Rapport de méthodes

Enquête suisse sur la structure des salaires

Programmes R pour l'intervalle de confiance de la médiane



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Département fédéral de l'intérieur DFI
Office fédéral de la statistique OFS

Neuchâtel, 2007

Ce rapport comporte deux parties. La première, plus mathématique, présente les calculs effectués dans le cadre de la LSE: la méthode suivie pour le calcul de la médiane, les différentes étapes nécessaires pour établir un intervalle de confiance à 95% et trois coefficients de variation ainsi que le traitement des domaines sont décrits. La deuxième partie décrit chaque élément du programme qui a été implémenté. Puis, les fonctions du package *survey* ayant un rapport avec les méthodes de la LSE sont analysées. Pour terminer, une comparaison des performances du programme et du package est présentée.

N° de commande: 338-0045
Commandes: 032 713 60 60
Fax: 032 713 60 61
E-Mail: order@bfs.admin.ch
Prix: gratuit
ISBN 978-3-303-00377-0

Statistik der Schweiz

Methodenbericht

Statistique de la Suisse

Rapport de méthodes

Enquête suisse sur la structure des salaires

Programmes R pour l'intervalle de confiance de la médiane

Auteurs **Jacques Ferrez et Monique Graf**
Office fédéral de la statistique

Editeur **Office fédéral de la statistique**

Neuchâtel, 2006

Préambule

Ce travail est le résultat d'un stage organisé par le service de méthodes statistiques. Le thème provient d'une question posée par Peter Moser, directeur de recherche à l'Office cantonal de statistique du canton de Zurich sur l'utilisation possible du logiciel R pour faire les calculs d'intervalle de confiance de la médiane des salaires pour l'enquête suisse sur la structure des salaires (Lohnstrukturerhebung, LSE), tels qu'ils sont réalisés à l'OFS à l'aide de SAS. Le logiciel R étant facile à installer et d'usage libre et gratuit, la question nous a semblé mériter une réponse circonstanciée. Les pages qui suivent décrivent la solution pratique proposée à l'usage des utilisateurs de la LSE, ainsi qu'une comparaison avec le package *survey* de T. Lumley.

Résumé

Ce rapport comporte deux parties. La première, plus mathématique, présente les calculs effectués dans le cadre de la LSE : la méthode suivie pour le calcul de la médiane, les différentes étapes nécessaires pour établir un intervalle de confiance à 95% et trois coefficients de variation ainsi que le traitement des domaines sont décrits. La deuxième partie décrit chaque élément du programme qui a été implémenté. Puis, les fonctions du package *survey* ayant un rapport avec les méthodes de la LSE sont analysées. Pour terminer, une comparaison des performances du programme et du package est présentée.

Mots-clé

rapport de méthodes; LSE; R; intervalle de confiance.

Complément d'information:	Monique Graf, tél. 032 713 66 15 Monique.Graf@bfs.admin.ch
Réalisation:	Service de méthodes statistiques, OFS
Diffusion:	Office fédéral de la statistique CH-2010 Neuchâtel Tél. 032 713 60 60 / Fax 032 713 60 61 Order@bfs.admin.ch
Internet:	http://www.statistik.admin.ch
Numéro de commande:	338-0045
Prix:	gratuit
Série:	Statistique de la Suisse
Domaine:	0 Bases statistiques et produits généraux
Langue du texte original:	Français
Graphisme/Layout:	OFS
Copyright:	OFS, Neuchâtel 2007 La reproduction est autorisée, sauf à des fins commerciales, si la source est mentionnée.
ISBN:	978-3-303-00377-0

Table des matières

Introduction 5

1 Contexte 5
1.1 L'enquête suisse sur la structure des salaires 5
1.2 Description des données 5
1.3 Notations 6

2 Formulation mathématique 6
2.1 Calcul de la médiane 6
2.2 Calcul de la précision 7
2.3 Equivalence des traitements des domaines 9
2.4 Remarque 10

3 Le programme 10
3.1 `computeQuantiles.R` 10
3.2 `statMed.R` 11
3.3 `lseComp.R` 14

4 Le package *survey* 15
4.1 Le design 15
4.2 Le calcul de la variance 17
4.3 Le calcul des statistiques 19
4.4 Autres fonctions 23
4.5 Application à la LSE 24

5 Performances 27

Conclusion 28

Annexes 29
A Mode d'emploi 29
B Code R 33

Références 40

Introduction

Les calculs d'intervalles de confiance de la médiane dans le cadre de la LSE (Enquête suisse sur la structure des salaires) sont effectués à l'aide de la macro `icmed02.sas` de Monique Graf. Pour certains offices cantonaux, disposant d'une partie seulement des données suisses, des calculs par le logiciel R [3] ont été envisagés. Dans le but de leur fournir les outils adéquats, nous avons étudié les possibilités qu'offre le logiciel R et le package *survey* de Thomas Lumley [4].

Les difficultés rencontrées lors de l'application de ce package ont motivé l'implémentation d'un programme spécifique. Ce rapport a pour but la description de la partie du package qui a été étudiée, ainsi que la présentation du programme qui a été implémenté.

Après une brève description de la LSE et des données sur lesquelles nous avons travaillé, nous détaillerons quelques aspects mathématiques (le calcul de la médiane, celui de la variance et le traitement des domaines) pour enfin passer aux aspects numériques, avec pour commencer le programme, puis la description de quelques fonctions du package *survey* de R.

Ce travail se base sur le rapport de méthodes de Monique Graf, *Enquête suisse sur la structure des salaires 2000. Plan d'échantillonnage, pondération et méthode d'estimation pour le secteur privé* ([1]). On y trouvera tous les détails relatifs à la LSE utiles à la compréhension de ce qui suit. Le package *survey* de R et sa documentation ont également été utilisés.

1 Contexte

1.1 L'enquête suisse sur la structure des salaires

Dans le cadre de la LSE, les entreprises suisses ont été réparties en strates selon la branche d'activité (classes NOGA 2), la taille (en fonction du nombre d'employés : de 3 à 19, de 20 à 49 et plus de 50) et la grande région (régions NUTS 2). Dans ces strates, un tirage aléatoire simple sans remise a été effectué, puis, dans chaque entreprise, des salaires ont été sélectionnés, à nouveau selon un tirage simple sans remise.

Le calcul de l'intervalle de confiance de la médiane commence évidemment par celui de la médiane. Ensuite, la méthode utilisée consiste à se placer sur l'échelle des pourcentages et à estimer la variance de l'image du salaire médian par la fonction de répartition. On tire de cette variance un écart-type, à partir duquel on calcule le coefficient de variation du percentile et l'intervalle de confiance à 95% de la médiane. On calcule enfin les coefficients de variation synthétiques à 68.62% et 95%. On trouvera davantage de détails à la section 2 et dans [1], chapitre 3.

1.2 Description des données

Les fichiers de données sur lesquels nous avons travaillé sont constitués de quinze colonnes

`identr`	identificateur de l'entreprise
`GESCHLE`	sexe du salarié
`ANFORNI`	niveau de qualifications requises pour le poste
`nog_2`	secteur d'activité (classe NOGA 2) de l'entreprise
`mbls`	montant du salaire mensuel brut standardisé
`gewibgrs`	poids d'extrapolation tenant compte du taux d'occupation
`ukto`	canton de l'entreprise

`stragrs`	identificateur de la strate
`nrep`	nombre d'entreprises répondant par strate
`ta3`	taille de l'entreprise
`grs`	région de l'entreprise
`gr`	grande région (NUTS 2) de l'entreprise
`anzlohn`	nombre de salaires communiqués par l'entreprise
`th`	taux de sondage effectif dans la strate
`thi`	taux de sondage dans l'entreprise.

On trouvera plus de détails à ce sujet dans [1]. Pour travailler avec le package *survey*, une colonne a été ajoutée, `NrSalaire`, constituée d'entiers de 1 à `n`, `n` étant le nombre de lignes du fichier. Cette colonne joue le rôle d'identificateur de salaire.

1.3 Notations

Dorénavant, on utilisera les notations suivantes

y_j	montant du j-ème salaire (`mbls`)
g_j	poids associé au j-ème salaire (`gewibgrs`)
s	échantillon
d	domaine
med	médiane pondérée des salaires
$\widehat{F}$	fonction de répartition empirique des salaires
m_{hi}	nombre de salaires sondés dans l'entreprise i (`anzlohn`)
$m_{d,hi}$	nombre de salaires sondés dans l'entreprise i dans le domaine
t_{hi}	taux de sondage de l'entreprise i (`thi`)
n_h	nombre d'entreprises sondées dans la strate h (`nrep`)
$n_{d,h}$	nombre d'entreprises sondées dans la strate h dans le domaine
t_h	taux de sondage effectif de la strate h (`th`)
B_{hi}	variance intra-entreprise de l'entreprise i de la strate h
B_h	somme des contributions intra-entreprises de la strate h
A_h	variance inter-entreprises de la strate h
e_j	$g_j(\mathbb{1}_{\{y_j \leqslant med\}} - 0.5)$, avec $\mathbb{1}_{\{y_j \leqslant med\}}$ = 1 si $y_j \leqslant med$ et 0 sinon
e_{hi}	somme des e_j au niveau de l'entreprise i de la strate h
$e_{d,hi}$	somme des e_j dans le domaine, au niveau de l'entreprise i
e_h	somme des e_j au niveau de la strate h
$e_{d,h}$	somme des e_j dans le domaine, au niveau de la strate h.

2 Formulation mathématique

2.1 Calcul de la médiane

Il existe plusieurs méthodes pour le calcul d'une médiane pondérée. Nous en détaillons deux : celle qui a été adoptée dans le cadre de la LSE et celle qui est utilisée par défaut dans le package *survey*.
Dans le contexte de la LSE, les poids (`gewibgrs`) sont d'abord classés dans l'ordre croissant des salaires (`mbls`). Les sommes partielles de ces poids sont ensuite calculées et divisées par la somme totale des poids. Enfin, si une des sommes partielles vaut exactement 0.5, alors la médiane est définie comme la moyenne du salaire correspondant et du salaire suivant ; si ce n'est pas le cas, la médiane est définie comme le salaire correspondant à la première somme

partielle qui dépasse 0.5

$$med = \begin{cases} \frac{1}{2}(y_{[i]} + y_{[i+1]}) & \text{si } \widehat{F}(y_{[i]}) = 0.5 \\ y_{[i+1]} & \text{si } \widehat{F}(y_{[i]}) < 0.5 < \widehat{F}(y_{[i+1]}), \end{cases}$$

où $\widehat{F}(y)$ désigne la somme partielle correspondant au salaire y. Cette définition correspond à celle qui est implémentée dans la procédure SAS Univariate. Dans le package *survey* (ou plus précisément dans la fonction `svyquantile()`, sur laquelle nous reviendrons à la section 4.3.4), par défaut la médiane est calculée comme suit. Les poids sont également classés dans l'ordre croissant des salaires et les sommes partielles calculées, puis divisées par la somme totale des poids. Mais cette fois-ci, une interpolation linéaire est effectuée avec comme premières coordonnées, les sommes partielles et comme deuxièmes coordonnées, les salaires. C'est en calculant l'image de 0.5 par cette fonction que la médiane est calculée.

$$med = \begin{cases} y_{[i]} & \text{si } \widehat{F}(y_{[i]}) = 0.5 \\ \alpha y_{[i]} + (1-\alpha) y_{[i+1]} & \text{si } \widehat{F}(y_{[i]}) < 0.5 < \widehat{F}(y_{[i+1]}), \end{cases}$$

avec

$$\alpha = \frac{\widehat{F}(y_{[i+1]}) - 0.5}{\widehat{F}(y_{[i+1]}) - \widehat{F}(y_{[i]})}.$$

Remarquons que ces deux méthodes ne se limitent pas au calcul de la médiane, mais permettent de calculer n'importe quel quantile q (il suffit de remplacer 0.5 par q). Les deux méthodes donnent presque toujours des résultats différents, la médiane calculée par la méthode de la LSE étant toujours supérieure ou égale à celle calculée par la méthode du package *survey* (m_survey $\leqslant$ m_LSE).

2.2 Calcul de la précision

Quatre indicateurs de précision sont calculés : un intervalle de confiance à 95%, des coefficients de variation synthétiques à 95% et à 68.62%, et le coefficient de variation du percentile. Pour cela, on commence par calculer la variance du percentile correspondant à la médiane (la méthode appliquée est celle de la linéarisation, pour davantage de détails voir [1], section 3.3 et [2], section 5.6), pour ensuite en déduire l'intervalle de confiance et les coefficients de variation.

2.2.1 Calcul de la variance

On définit, pour chaque salaire j, la variable e_j

$$e_j = g_j \left(\mathbb{1}_{\{y_j \leqslant med\}} - 0.5 \right),$$

où y_j représente le j-ème salaire, g_j, son poids associé et med, la médiane. Soit m_{hi} le nombre de salaires de l'entreprise i dans l'échantillon et $m_{d,hi}$ le nombre de salaires de l'entreprise i qui se trouvent dans l'échantillon et dans le domaine d dont on calcule la médiane (il se peut que les salaires d'une strate ou d'une entreprise ne soient qu'en partie dans le domaine d'étude ; nous reviendrons sur ce cas à la section 2.3). De manière analogue, soit n_h le nombre d'entreprises de la strate h dans l'échantillon et $n_{d,h}$, le nombre d'entreprises de la strate h dans l'échantillon et dans le domaine d.

Pour calculer la variance globale, il est nécessaire d'établir, d'une part, la variance des salaires dans les entreprises, ou variance intra-entreprises, et, d'autre part, la variance des salaires entre les entreprises, ou variance inter-entreprises.

D'après ([1], p. 28), la variance intra-entreprise de l'entreprise i de la strate h, notée B_{hi}, se calcule ainsi

$$B_{hi} = \frac{m_{d,hi}-1}{m_{hi}-1}\text{Var}[e_j] + \frac{m_{d,hi}}{m_{hi}-1}\left(1-\frac{m_{d,hi}}{m_{hi}}\right)\left(\frac{e_{d,hi}}{m_{d,hi}}\right)^2,$$

où $e_{d,hi}$ désigne la somme des e_j correspondant aux salaires de l'entreprise i dans le domaine d et où la variance Var$[e_j]$ est prise sur les indices j correspondant aux salaires de l'entreprise i (notons que $e_{d,hi}/m_{d,hi}$ correspond alors à la moyenne des e_j). La contribution de l'entreprise i à la variance totale est donnée par

$$\begin{aligned} B'_{hi} &= m_{hi}(1-t_{hi})B_{hi} \\ &= m_{hi}(1-t_{hi})\left[\frac{m_{d,hi}-1}{m_{hi}-1}\text{Var}[e_j] + \frac{m_{d,hi}}{m_{hi}-1}\left(1-\frac{m_{d,hi}}{m_{hi}}\right)\left(\frac{e_{d,hi}}{m_{d,hi}}\right)^2\right]. \end{aligned}$$

Ces contributions sont sommées afin d'obtenir B_h, la somme des contributions intra-entreprises au niveau de la strate

$$B_h = \sum B'_{hi} = \sum m_{hi}(1-t_{hi})B_{hi}.$$

Passons à présent à la variance inter-entreprises dans une strate, notée A_h. Celle-ci se calcule de manière analogue (voir [1], p. 28)

$$A_h = \frac{n_{d,h}-1}{n_h-1}\text{Var}\left[e_{hi}\right] + \frac{n_{d,h}}{n_h-1}\left(1-\frac{n_{d,h}}{n_h}\right)\left(\frac{e_{d,h}}{n_{d,h}}\right)^2,$$

où $e_{d,h}$ représente la somme des e_j correspondant aux salaires de la strate h (à nouveau, $e_{d,h}/n_{d,h}$ correspond à la moyenne des e_j). La contribution inter-entreprises de la strate h à la variance totale est donnée par

$$\begin{aligned} A'_h &= n_h(1-t_h)A_h \\ &= n_h(1-t_h)\left[\frac{n_{d,h}-1}{n_h-1}\text{Var}\left[e_{hi}\right] + \frac{n_{d,h}}{n_h-1}\left(1-\frac{n_{d,h}}{n_h}\right)\left(\frac{e_{d,h}}{n_{d,h}}\right)^2\right]. \end{aligned}$$

Pour obtenir la contribution totale de la strate h à la variance, on somme les contributions inter et intra-entreprises

$$V_{2st,h} = A'_h + t_h B_h$$

(voir [1], section 3.3.4 ou [2], résultat 4.3.1). Enfin, pour la variance globale, on divise la somme des variances des strates par le carré de la somme totale des poids

$$SV_{2st} = \frac{\sum V_{2st,h}}{\left(\sum g_j\right)^2}.$$

2.2.2 Intervalle de confiance et coefficients de variation

De la variance SV_{2st}, on tire l'écart-type du percentile

$$sep = \sqrt{SV_{2st}},$$

qui va nous permettre d'établir l'intervalle de confiance à 95% et les trois coefficients de variation. Pour obtenir le coefficient de variation du percentile, noté CV_{perc}, on divise simplement l'écart-type par 0.5, la valeur théorique du percentile de la médiane

$$CV_{perc} = 100 \cdot \frac{sep}{0.5}$$

(les différents coefficients de variation seront exprimés en pourcents). Pour obtenir l'intervalle de confiance à 95%, noté $[b_i, b_s]$, on calcule un intervalle de confiance sur l'échelle des pourcentages et on en prend la préimage par la fonction de répartition empirique

$$[b_i, b_s] = \widehat{F}^{-1}\left[0.5 - 1.96 \cdot sep, 0.5 + 1.96 \cdot sep\right],$$

où $\widehat{F}^{-1}$ désigne la fonction de répartition inverse. Pour le coefficient de variation synthétique à 95%, noté CV_{syn95}, on prend le plus grand demi-intervalle que l'on divise par $1.96 \cdot med$

$$CV_{syn95} = 100 \cdot \frac{\max(med - b_i, b_s - med)}{1.96 \cdot med}.$$

Enfin, pour le coefficient de variation synthétique à 68.62%, que l'on note CV_{syn}, on calcule l'intervalle suivant

$$[c_i, c_s] = \widehat{F}^{-1}\left[0.5 - sep, 0.5 + sep\right]$$

et on procède comme pour CV_{syn95}

$$CV_{syn} = 100 \cdot \frac{\max(med - c_i, c_s - med)}{med}$$

(pour plus de détails, voir [1], chapitre 3).

2.3 Equivalence des traitements des domaines

Les variances que nous avons calculées jusqu'à présent, A_h et B_{hi}, comportent une somme de deux termes dont le premier dépend d'une variance empirique et le deuxième du carré d'une somme (qui est en fait le carré d'une moyenne). Le deuxième terme correspond au traitement des domaines qui ne sont pas constitués de strates entières (c'est le cas, par exemple, si on ne considère qu'un seul sexe ou seulement certains niveaux de qualifications). On constate en effet, dans le cas de A_h, un facteur $1 - n_{d,h}/n_h$ qui est nul si les salaires échantillonnés de la strate sont tous considérés, et un facteur $(n_{d,h} - 1)/(n_d - 1)$ qui vaudra 1 dans le même cas (la situation est analogue pour B_{hi}).

Il existe une autre méthode pour calculer ces variances. Soit s l'échantillon de données et n_s sa taille. Considérons un domaine d, on a alors un sous-échantillon $s_d = s \cap d$, de taille n_{s_d}. Posons

$$S^2_{ys_d} = \frac{1}{n_{s_d} - 1} \sum_{s_d} \left(y_k - \overline{y}_{s_d}\right)^2,$$

où $\overline{y}_{s_d}$ représente la moyenne des $y_k \in s_d$. L'estimation de la variance donnée par la formule

$$\widehat{V} = n_s(1 - f)\left[\frac{n_{s_d} - 1}{n_s - 1} S^2_{ys_d} + \frac{n_{s_d}}{n_s - 1}\left(1 - \frac{n_{s_d}}{n_s}\right)\overline{y}^2_{s_d}\right]$$

([2], section 10.3, exemple 10.3.1), où f représente le taux de sondage, peut être directement obtenue par le calcul de la variance des données auxquelles on ajoute $n_s - n_{s_d}$ zéros (on a alors à nouveau n_s données). Notons s^* l'échantillon auquel on a ajouté les zéros, $\overline{y}_{s^*}$ la moyenne des $y_k \in s^*$ et $S^2_{ys^*}$ la variance empirique de ces nouvelles données. On a alors

$$
\begin{aligned}
S^2_{ys^*} &= \frac{1}{n_s-1}\sum_{s^*}(y_k-\overline{y}_{s^*})^2 \\
&= \frac{1}{n_s-1}\sum_{s^*}\left(y_k-\frac{n_{s_d}}{n_s}\overline{y}_{s_d}\right)^2 \\
&= \frac{1}{n_s-1}\left[\sum_{s_d}\left(y_k-\frac{n_{s_d}}{n_s}\overline{y}_{s_d}\right)^2+\sum_{s*-s_d}\left(y_k-\frac{n_{s_d}}{n_s}\overline{y}_{s_d}\right)^2\right] \\
&= \frac{1}{n_s-1}\left[\sum_{s_d}\left(y_k-\overline{y}_{s_d}+\overline{y}_{s_d}-\frac{n_{s_d}}{n_s}\overline{y}_{s_d}\right)^2+(n_s-n_{s_d})\frac{n^2_{s_d}}{n^2_s}\overline{y}^2_{s_d}\right] \\
&= \frac{1}{n_s-1}\left[\sum_{s_d}(y_k-\overline{y}_{s_d})^2+n_{s_d}\overline{y}^2_{s_d}\left(1-\frac{n_{s_d}}{n_s}\right)^2+(n_s-n_{s_d})\frac{n^2_{s_d}}{n^2_s}\overline{y}^2_{s_d}\right] \\
&= \frac{1}{n_s-1}\left[(n_{s_d}-1)S^2_{ys_d}+\overline{y}^2_{s_d}\left(n_{s_d}\left(\frac{n_s-n_{s_d}}{n_s}\right)^2+(n_s-n_{s_d})\frac{n^2_{s_d}}{n^2_s}\right)\right] \\
&= \frac{n_{s_d}-1}{n_s-1}S^2_{ys_d}+\frac{1}{n_s-1}\overline{y}^2_{s_d}\frac{n_{s_d}(n_s-n_{s_d})}{n^2_s}((n_s-n_{s_d})+n_{s_d}) \\
&= \frac{n_{s_d}-1}{n_s-1}S^2_{ys_d}+\frac{n_{s_d}}{n_s-1}\left(1-\frac{n_{s_d}}{n_s}\right)\overline{y}^2_{s_d}
\end{aligned}
$$

et les deux méthodes de calcul donnent bien la même estimation de la variance.

2.4 Remarque

Dans certaines entreprises ou dans certaines strates, il se peut que le manque de données rende le calcul de la variance impossible. En ce qui concerne l'expression B'_{hi}, elle est fixée à 0 si $t_{hi} = 1$ (échantillon exhaustif), pour supprimer des valeurs manquantes qui pourraient apparaître si $m_{hi} = 1$ (il y aurait division par $m_{hi} - 1 = 0$) ou si $m_{d,hi} = 1$ (il n'y aurait qu'un seul salaire et la variance empirique ne pourrait pas être calculée). Lors du calcul de B_h, les B_{hi} qui n'ont pas pu être calculés (si $t_{hi} < 1$ et si $m_{hi} = 1$ ou si $m_{d,hi} = 1$) ne sont pas pris en compte. Dans le cas de la variance A_h, on procède à une imputation (voir la section 3.2.3) avant de poser $A_h = 0$ si $t_h = 1$ (échantillon exhaustif) et $n_{d,h} = 1$ (le cas $t_h = 1$ et $n_h = 1$ est aussi traité, car si $n_h = 1$, alors $n_{d,h} = 1$).

3 Le programme

Le programme qui a été implémenté pour le calcul des intervalles de confiance dans le cadre de la LSE est constitué de trois fonctions : `computeQuantiles.R`, qui calcule des quantiles pondérés, `statMed.R`, qui calcule un intervalle de confiance et des coefficients de variation, et `lseComp.R`, qui permet le calcul de ces statistiques pour différents domaines de salaires.

3.1 `computeQuantiles.R`

La fonction `computeQuantiles()` calcule des quantiles pondérés. Elle prend trois arguments

- `xx` données dont on calcule les quantiles (`mbls`)
- `ww` poids (`gewibgrs`, par défaut fixés à 1)
- `qq` quantiles à calculer (par défaut fixé à 0.5).

Si les poids ne sont pas spécifiés, la fonction renvoie des quantiles non-pondérés. Si au contraire, les poids ont bien été mentionnés, la méthode utilisée dans le contexte de la LSE (et décrite à la section 2.1) est appliquée dans une boucle `for` qui est effectuée pour chaque quantile à calculer. Les valeurs obtenues sont alors retournées.
Si `data` est un fichier contenant, par exemple, les données du secteur secondaire de la grande région 4 pour la LSE02, alors la commande

```
computeQuantiles(data$mbls, data$gewibgrs, 0.5)
```

renverra la médiane `5953`.

3.2 `statMed.R`

La fonction `statMed()` calcule un intervalle de confiance à 95%, des coefficients de variation synthétiques à 95% et 68.62% et le coefficient de variation du percentile. Elle prend onze arguments

`x`	données dont on calcule la médiane (`mbls`)
`strata`	identificateurs de strates (`stragrs`)
`psu`	identificateurs d'entreprises (`identr`)
`nh`	nombre d'entreprises échantillonnées dans la strate h (`nrep`)
`th`	taux effectif d'échantillonnage de la strate h (`th`)
`Nh`	nombre total d'entreprises dans la strate h (`nrep/th`)
`mhi`	nombre de salaires échantillonnés dans l'entr. i (`anzlohn`)
`thi`	taux d'échantillonnage de l'entreprise i (`thi`)
`Mhi`	nombre total de salaires dans l'entreprise i (`anzlohn/thi`)
`weights`	poids à utiliser lors des calculs (`gewibgrs`)
`crit`	critère à utiliser lors de l'imputation de la variance.

Il n'est pas nécessaire que tous ces arguments soient spécifiés lors de l'appel de `statMed()`. Par conséquent, ils ont une valeur par défaut fixée à `NULL`, excepté `x` (qui est évidemment indispensable). Le code se divise en quatre parties. Pour commencer, les données sont testées et mises sous une forme cohérente, puis le calcul de la médiane, celui des variances ainsi que la procédure d'imputation sont effectués. Enfin, les indicateurs de précision sont calculés et retournés.
Remarquons que les données fournies par la section LOHN de l'OFS ne contiennent pas toutes les variables listées ci-dessus, il s'agit donc de les calculer. A cet effet, on trouvera un script au début de l'annexe 5.

3.2.1 Tests

Les données sont contrôlées, d'abord au niveau global, puis au niveau des entreprises et enfin au niveau des salaires. Si les identificateurs de strates (`strata`) n'ont pas été spécifiés, il est supposé qu'il n'y a qu'une seule strate. Le même processus est appliqué aux identificateurs d'entreprises (`psu`).
Si le nombre d'entreprises échantillonnées dans la strate n'a pas été spécifié, on suppose que le domaine est constitué de strates entières : les entreprises sont comptées dans chaque strate et le résultat obtenu est défini comme le nombre d'entreprises échantillonnées dans la strate. Dans le cas où ni le nombre total d'entreprises (`Nh`), ni le taux de sondage (`th`) n'ont été spécifiés, le taux est supposé égal à 1 et `Nh` égal à `nh`. Si un seul de ces paramètres a été

spécifié, (cas standard), l'autre paramètre en est déduit. Une procédure de test en tout point identique est appliquée au niveau des salaires.
Enfin, les poids `weights` sont testés. S'ils n'ont pas été spécifiés, ils sont posés égaux à `1/(th*thi)`, ce qui correspond aux poids d'échantillonnage.

3.2.2 Médiane et variance

A l'aide de la fonction `computeQuantiles()`, on calcule la médiane, notée `med`. En vue du calcul de la variance, les valeurs de `zhij` et `ej` sont ensuite calculées pour chaque salaire

```
zhij  <-  1 * (x <= med)
ej    <-  weights * (zhij - 0.5).
```

Les variances intra-entreprises `Bhi` de chaque entreprise sont ensuite établies. On part de la variance empirique des `ej` pour chaque entreprise et on calcule `Bhi` (pour cela, il faut encore calculer $m_{d,hi}$, le nombre de salaires dans le domaine, noté `NDhi` dans le code)

```
Bhi  <-  ((NDhi-1)*Bhi + NDhi*(1-NDhi/mhi)*(ehi/NDhi)^2)
         *(1-thi)*mhi/(mhi-1).
```

Les `Bhi` correspondant à des `thi` qui valent 1 sont posés égaux à 0 (comme nous l'avons vu à la section 2.4).
Les sommes des contributions intra-entreprises au niveau des strates, B_h, sont ensuite calculées. S'il n'y a qu'une seule entreprise dans la strate, et donc un seul `Bhi`, c'est ce `Bhi` qui est pris en compte, même s'il a pour valeur `NA` (la strate n'aura dans ce cas aucune contribution à la variance globale). Si, au contraire, la strate comporte plusieurs entreprises, et donc plusieurs `Bhi`, `Bh` prend la valeur de la somme des `Bhi` différents de `NA`.
Un processus similaire est appliqué pour le cas de la variance inter-entreprises, notée `Ah`. On part de la variance empirique des sommes de `ej` pour chaque entreprise (`ehi`), on calcule $n_{d,h}$, le nombre d'entreprises dans le domaine (`ne`) et on calcule les variances `Ah`

```
Ah  <-  ((ne-1)*Ah + ne*(1-ne/nh)*(eh/ne)^2) / (nh-1).
```

C'est à ce stade qu'intervient l'imputation, si elle est nécessaire.

3.2.3 Imputation

Si le critère d'imputation a été spécifié, si certains `Ah` ont pour valeur `NA` et si d'autres ont pu être calculés, alors on procède à l'imputation, selon le critère passé en argument lors de l'appel de `statMed()`. Dans le cadre de la LSE, l'imputation se fait sur les strates de la même grande région et de la même classe NOGA 2.
Pour commencer, une variance relative, notée `Ahrel`, est calculée : pour chaque strate, `Ah` est divisé par le carré de la somme des poids de la strate (noté `svh`)

```
Ahrel <- Ah / svh^2,
```

ensuite de quoi la moyenne des `Ahrel` différents de `NA` est calculée pour chaque valeur du critère d'imputation `crit` ; on la note `m_Ahrel`. Enfin, pour chacune des strates ne contenant qu'une entreprise (`ne==1`), `Ah` prend la valeur de `m_Ahrel` qui lui correspond selon le critère `crit`, multipliée par le carré de la somme des poids de sa strate, `svh^2`

```
Ah <- svh^2 * m_Ahrel.
```

Enfin, si `ne==1` et `th==1`, alors `Ah` est posé égal à zéro. Finalement, tous les `Ah` sont multipliés par `nh*(1-th)`.

3.2.4 Valeurs retournées

Pour terminer, la variance de chaque strate, `V2sth`, est calculée, ainsi que la variance globale `SV2st`, dont on déduit l'écart-type `sep`

```
V2sth  <-  Ah + th*Bh
SV2st  <-  sum(V2sth)/sum(svh)^2
sep    <-  sqrt(SV2st)
```

(notons que lors du calcul de `SV2st`, les strates pour lesquelles `V2sth=NA` ne sont pas prises en compte). De cet écart-type et grâce à la fonction `computeQuantiles()`, on tire le coefficient de variaton du percentile, `CVperc`, les bornes d'un intervalle de confiance à 95%, `l.limit` et `u.limit`, ainsi que celles d'un intervalle de confiance à 68.62%, `cl` et `cu`

```
CVperc   <-  100 * sep/0.5
l.limit  <-  computeQuantiles(x, weights, 0.5 - 1.96 * sep)
u.limit  <-  computeQuantiles(x, weights, 0.5 + 1.96 * sep)
cl       <-  computeQuantiles(x, weights, 0.5 - sep)
cu       <-  computeQuantiles(x, weights, 0.5 + sep).
```

Enfin, à partir de ces intervalles de confiance, on calcule les coefficients de variation synthétiques à 95% et à 68.62%, `cv_s95` et `cv_s`

```
cv_s95  <-  100 * max(med-l.limit, u.limit-med)/(1.96*med)
cv_s    <-  100 * max(med-cl, cu-med)/med.
```

Les valeurs retournées sont les bornes de l'intervalle de confiance à 95% (`l.limit` et `u.limit`), la médiane (`med`), les coefficients de variation synthétiques à 95% et à 68.62% (`cv_s95` et `cv_s`), le coefficient de variation du percentile (`CVperc`), le nombre de strates (`Nstrata`), le nombre d'entreprises (`Npsu`) et, enfin, le nombre de salaires (`Nssu`).

3.2.5 `statMedM.R`

Dans certains cas, il peut se révéler utile d'identifier les composantes inter-entreprises et intra-entreprise de la variance globale. La contribution inter-entreprises est aisément calculable. Il suffit de spécifier, lors de l'appel de la fonction `statMed()`, le paramètre `thi=1`. Les variances `Bhi` seront donc toutes fixées à 0 et la variance globale ne tiendra compte que des variances inter-entreprises. Afin d'être en mesure d'identifier la part intra-entreprises de la variance globale, une version modifiée de `statMed.R` a été mise au point, `statMedM.R`, qui permet de passer en argument la médiane par rapport à laquelle la variance doit être calculée (il ne suffit pas de faire le calcul avec `statMed()` sur les données de chaque entreprise séparément, car c'est la variance de la médiane de l'entreprise qui serait calculée et non pas la variance de la médiane globale dans l'entreprise).
La seule différence avec `statMed()` réside dans le fait que la médiane n'est pas calculée dans le programme. En revanche, il faut passer en argument une valeur pour `med`, la médiane.

3.2.6 Exemple

Si `data` est un fichier contenant les données du secteur secondaire de la grande région 4 pour la LSE02, alors les commandes

```
critgrnog_2 <- as.numeric(paste(data$gr, data$nog_2, sep=""))
```

et

```
statMed(data$mbls, strata=data$stragrs, psu=data$identr, nh=data$nrep,
th=data$th, mhi=data$anzlohn, thi=data$thi, weights=data$gewibgrs,
crit=critgrnog_2)
```

renverront les valeurs du tableau ci-dessous.

```
l.limit  u.limit  median     cv_s95       cv_s    CVperc
   5890     6004    5953  0.5399438  0.5711406  1.440844
Nstrata     Npsu    Nssu
     51     1829   57266
```

3.3 `lseComp.R`

La fonction `lseComp()` calcule une médiane pondérée et plusieurs statistiques sur sa précision. Elle prend trois arguments

`data`	données à traiter
`...`	données en fonction desquelles les résultats seront détaillés
`noga_spec`	vecteur de listes de classes NOGA 2.

Le fichier `data` doit contenir les colonnes suivantes : `mbls`, `stragrs`, `identr`, `nrep`, `th`, `anzlohn`, `thi`, `gewibgrs`, `gr` et `nog_2`. Si un vecteur de listes est spécifié pour `noga_spec`, le calcul sera effectué pour chacune de ces listes.

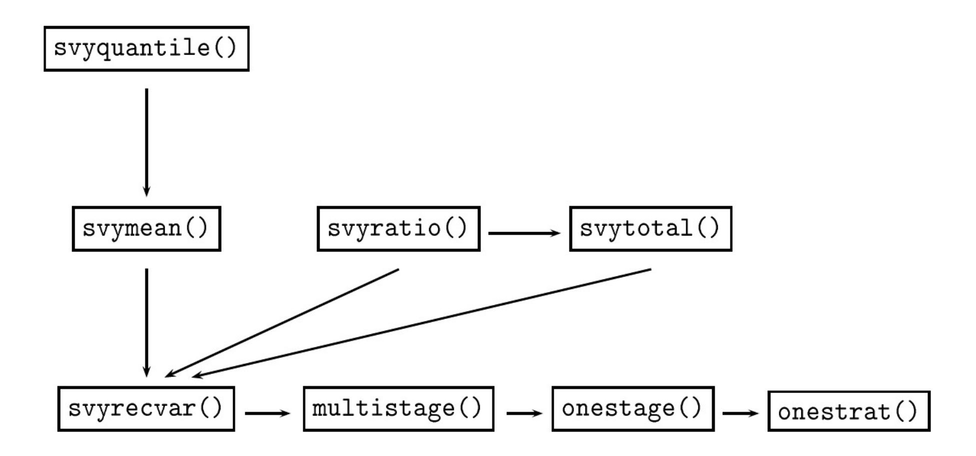
La fonction ne s'effectue que si le nombre de lignes de `data` est positif. Si des données sont spécifiées, (hormis `data` et `noga_spec`), la fonction s'applique elle même aux sous-ensembles de `data` correspondant à la première de ces données, avec les autres données en paramètre. Ensuite, de manière similaire, si une valeur a été spécifiée pour `noga_spec`, la fonction s'appelle elle-même pour chaque liste contenue dans `noga_spec`, en adaptant `data` et `noga_spec`. Enfin, si aucun de ces paramètres n'a été spécifié, la fonction met en mémoire les valeurs de `gr`, `nog_2`, `GESCHLE`, `ANFORNI` et `ta3` afin de les afficher avec les statistiques qui seront calculées. Le critère d'imputation est construit par concaténation des colonnes `gr` et `nog_2` puis stocké dans la variable `critgrnog_2`. Pour finir, la fonction `statMed()` est appelée

```
statMed(x=data$mbls, strata=data$stragrs, psu=data$identr, nh=data$nrep,
th=data$th, mhi=data$anzlohn, thi=data$thi, weights=data$gewibgrs,
crit=critgrnog_2).
```

Le résultat est retourné, ainsi que les différentes valeurs qui avaient été mises en mémoire.

Si le fichier `data` contient les données relatives à la LSE02 pour le secteur secondaire de la grande région 4, alors le code `lseComp(data, ~GESCHLE)` affichera le résultat du tableau ci-dessous.

```
       gr       noga   GESCHLE   ANFORNI    ta3   l.limit   u.limit   median
        4      10-45       1-2       1-4    1-3      5890      6004     5953
        4      10-45         1       1-4    1-3      6066      6206     6131
        4      10-45         2       1-4    1-3      4970      5143     5057
   cv_s95       cv_s    CVperc   Nstrata   Npsu      Nssu
0.5399438  0.5711406  1.440844        51   1829     57266
0.6241283  0.7176643  1.620436        51   1739     43689
0.8777488  1.0480522  1.994509        51   1496     13577
```



4 Le package *survey*

Le package *survey* de Thomas Lumley [4] est destiné aux calculs de statistiques pour des enquêtes stratifiées à plusieurs niveaux. Le nombre de niveaux n'est en théorie pas limité. Les fonctions du package peuvent en général traiter plusieurs variables simultanément. Certaines fonctions permettent de calculer l'effet du design, d'effectuer une poststratification ou encore une calibration. Les poids pris en compte sont généraux, il ne doit pas forcément s'agir des poids d'extrapolation. Par contre, aucune imputation n'est possible.
L'utilisation du package *survey* nécessite deux étapes. Il faut commencer par spécifier le design de l'enquête à l'aide de la fonction `svydesign()`, pour ensuite lancer le calcul désiré, à l'aide de l'une des fonctions `svyquantile()`, `svymean()`, `svyratio()` ou `svytotal()` (le package *survey* comporte d'autres fonctions, mais elles ne seront pas traitées dans ce rapport). Le code de ces fonctions est disponible à l'adresse http://cran.ch.r-project.org/src/contrib/Descriptions/survey.html, dans le fichier `Multistage.R` qui se trouve dans l'archive `survey_3.6-5.tar.gz`. Le code de `svytotal()` est disponible via la commande

```
survey : : :svytotal.survey.design2
```

(cette commande fonctionne aussi pour `svyratio()` et `svymean()`). Pour `svyquantile()`, il faut utiliser la commande `getS3method("svyquantile", "survey.design")`. Notons que ces fonctions sont décrites dans le manuel de référence du package ([4]) qui est disponible à l'adresse citée plus haut. Ainsi, nous ne détaillerons que les aspects qui nous semblent importants.
Les fonctions `svymean()`, `svyratio()` et `svytotal()` font toutes appel à `svyrecvar()` pour le calcul de la variance. Cette dernière appelle la fonction `multistage()`, qui appelle `onestage()` qui, enfin, appelle `onestrat()`. Notons encore que `svyquantile()` appelle `svymean()` pour le calcul de la variance et `svyratio()` appelle `svytotal()` pour le calcul du ratio.

4.1 Le design

4.1.1 `svydesign()`

La fonction `svydesign()` permet de spécifier le design d'une enquête. Son rôle se résout essentiellement à tester les données et à les mettre sous une forme adéquate pour la suite des calculs. Elle prend neuf paramètres

`ids`	identificateurs des unités d'échantillonnage
`probs`	probabilités d'échantillonnage
`strata`	identificateurs des strates
`variables`	variables mesurées lors de l'enquête
`fpc`	taux de sondage ou tailles des populations totales
`data`	données
`nest`	si `TRUE`, empêche les PSU d'être dans plusieurs strates
`check.strata`	si `TRUE`, vérifie que les PSU sont dans une strate
`weights`	poids d'échantillonnage.

Pour commencer, le paramètre `ids`, s'il n'a pas été spécifié, est posé égal à `1:n`, avec `n` le nombre de lignes de `data`. Ensuite, si les arguments `probs` et `weights` ont tous deux été spécifiés, le programme s'arrête

```
stop("Can't specify both sampling weights and probabilities").
```

Si, par contre, `weights` a été spécifié, mais pas `probs`, ce dernier est posé égal à l'inverse de `weights`. La variable booléenne `has.strata` est ensuite définie (`TRUE` si des strates ont été mentionnées, `FALSE` sinon). Si aucune valeur pour le paramètre `variables` n'a été précisé, il prend la valeur de `data`. Ensuite, afin que les unités de sondage du niveau `n+1` ne soient pas à cheval sur plusieurs unités de sondage du niveau `n`, les identificateurs des unités du niveau `n+1` sont renommées en $U_n.U_{n+1}$, où U_n et U_{n+1} désignent les identificateurs des unités de niveau `n` et `n+1`.
Après cela, `N-1` colonnes (s'il y a `N` niveaux d'échantillonnage) sont ajoutées à `strata`. Ce sont des identificateurs de sous-strates, composés des identificateurs des strates d'une part et des identificateurs des unités d'échantillonnage (sauf le dernier) d'autre part (ces colonnes sont utilisées dans la fonction `as.fpc()`). La fonction `as.fpc()` est ensuite appelée

```
fpc <- as.fpc(fpc, strata, ids)
```

(nous verrons ce qu'elle fait dans la section suivante). Dans le cas où ni `probs`, ni `weights` n'ont été spécifié, `fpc` est utilisé pour calculer des probabilités d'échantillonnage : elles sont fixées égales à `fpc$sampsize/fpc$popsize`. Si les tailles de populations totales ne sont pas disponibles non plus, `probs` est fixé à 1. Enfin, les valeurs suivantes sont retournées : `ids`, `strata`, `has.strata`, `allprob`, qui correspond à `probs`, `prob`, qui contient les produits par ligne de `probs` (si `probs` a plusieurs colonnes), `variables` et `fpc`.
Désormais, on parlera de strates pour désigner à la fois les strates et les sous-strates.

4.1.2 `as.fpc()`

La fonction `as.fpc()` renvoie une liste de deux éléments : `popsize` (les tailles des populations totales) et `sampsize` (les tailles des populations échantillonnées). Elle prend trois arguments

`df`	taux de sondage ou tailles des populations totales
`strata`	identificateurs des strates
`ids`	identificateurs des unités d'échantillonnage.

C'est d'abord `sampsize` qui est construit. Pour chaque niveau d'échantillonnage et dans chaque strate, les différentes valeurs de l'identificateur de l'unité de sondage sont comptées et stockées dans une matrice nommée `sampsize`, de même taille que `ids`. Si le paramètre `df` n'a pas été mentionné, la fonction s'arrête là.
Sinon, les données passées dans le paramètre `df` sont testées. Si `df` contient des valeurs strictement inférieures à un et d'autres valeurs strictement supérieures à un, ou si toutes les valeurs sont égales à un, la fonction s'arrête

```
stop("Must have all fpc>=1 or all fpc<=1").
```

Si `df` comprend des valeurs strictement supérieures à un, `popsize` prend la valeur de `df`, sinon, `sampsize/df`.
Pour terminer, le programme vérifie que les valeurs de `popsize` sont bien constantes dans les strates à chaque niveau.

4.2 Le calcul de la variance

Le package *survey* décompose le traitement des données en plusieurs parties. Comme nous venons de le voir, la première d'entre elles consiste à spécifier le design de l'enquête. Ensuite, il s'agit de calculer les statistiques voulues ainsi que leurs variances, c'est ce que font les fonctions `onestrat()`, `onestage()`, `mulstistage()` et `svyrecvar()`.

4.2.1 `onestrat()`

La fonction `onestrat()` calcule la variance dans une unité d'échantillonnage. Elle prend huit arguments

`x`	variable dont on calcule la variance
`cluster`	identificateurs des unités d'échantillonnage
`nPSU`	tailles originales des populations échantillonnées
`fpc`	tailles des populations totales
`lonely.psu`	définit le traitement des unités avec un seul élément
`stratum`	identificateurs des strates
`stage`	niveau de sondage en traitement
`cal`	contient les détails de la calibration.

La valeur par défaut du paramètre `lonely.psu` est `"fail"` (dans les options globales de R), ce qui correspond à l'envoi d'un message d'erreur et à l'arrêt du programme lorsqu'une strate ne contient qu'un élément. La valeur qui correspond le mieux au contexte de la LSE est `"remove"`, parce que les entreprises avec un seul salaire voient leurs variances B_{hi} fixées à zéro (si $t_{hi} = 1$) ou négligées. Il faut donc modifier la valeur de l'option dans R.
Le paramètre `stage` n'a que peu d'importance, il n'apparaît que dans les messages d'erreur affichés, soit dans le cas du traitement d'un domaine, lorsqu'il ne reste qu'un élément dans une strate et qu'à l'origine elle en contenait plusieurs, soit dans le cas où la taille originale d'une population échantillonnée vaut un.
Dans le contexte de la LSE, le paramètre `cal` ne revêt aucune importance, puisque aucune calibration n'est effectuée.
Passons à présent à l'analyse du code. Pour commencer, une valeur `f` est calculée. Elle correspond au facteur $1 - t_h$ (ou $1 - t_{hi}$) que nous avons vu à la section 2.2.1. Ensuite, le facteur `scale` est calculé. Il correspond, lui, à $(1 - t_h)n_h/(n_h - 1)$. Si `f=0`, une matrice de zéros est renvoyée.
Les données `x` sont alors sommées par valeur de `cluster` et `nsubset` stocke le nombre de `cluster`. Si `nsubset` est inférieur à `nPSU`, `x` est complété par des zéros (cela correspond à un domaine et le traitement appliqué a été décrit à la section 2.3).
La fonction calcule ensuite les écarts à la moyenne de `x` (pour être précis, ce sont les écarts aux moyennes par colonnes qui sont calculés ; dans le cadre de la LSE, `x` ne contient qu'une seule colonne).
Si, dans l'échantillon original, la strate contient plus d'un élément, mais pour cause de traitement d'un domaine, il n'en reste qu'un, un message d'erreur est envoyé.

Pour terminer, la valeur

$$\texttt{scale*crossprod(x)} = \texttt{scale} \cdot \sum \texttt{(x}_\texttt{i}\texttt{-}\overline{\texttt{x}}\texttt{)\^{}2}$$

est renvoyée. On constate que, excepté pour l'imputation, les calculs effectués par `onestrat()` correspondent à la théorie appliquée dans le cadre de la LSE, pour autant que les paramètres soient spécifiés correctement, c'est-à-dire

`x`	`gewibgrs*((mbls<=med)-0.5)`
`cluster`	`NrSalaire` ou `identr`
`nPSU`	tailles originales des échantillons (`nrep` ou `anzlohn`)
`fpc`	tailles des populations totales
`lonely.psu`	`"remove"`
`stratum`	identificateurs de strates.

`NrSalaire` correspond en fait à un vecteur `1:n`, avec `n` le nombre de salaires. Cette variable joue le rôle d'identificateur de salaire. `fpc` correspond à `nrep/th` et `anzlohn/thi`. Les différents paramètres sont calculés par la fonction `svydesign()`. Les valeurs de `stage` et de `cal` ne sont pas importantes.

4.2.2 `onestage()`

La fonction `onestage()` ne fait qu'appeler `onestrat()` pour chaque unité d'échantillonnage d'un niveau de sondage et renvoyer la somme des variances obtenues. Comme `onestrat()`, elle prend huit arguments

`x`	variable dont on calcule la variance
`strata`	identificateurs des strates
`clusters`	identificateurs des unités d'échantillonnage
`nPSU`	tailles originales des populations échantillonnées
`fpc`	tailles des populations totales
`lonely.psu`	définit le traitement des strates avec un seul élément
`stage`	niveau de sondage en traitement
`cal`	contient les détails de la calibration.

Tous ces paramètres sont passés tels quels lors de l'appel de `onestrat()`, au détail près que seules les lignes correspondant à l'unité en traitement sont sélectionnées.

4.2.3 `multistage()`

La fonction `multistage()`, elle, applique `onestage()` aux différents niveaux d'échantillonnage. Elle prend neuf arguments

`x`	variable dont on calcule la variance
`clusters`	identificateurs des unités d'échantillonnage
`stratas`	identificateurs des strates
`nPSUs`	tailles originales des populations échantillonnées
`fpcs`	tailles des populations totales
`lonely.psu`	définit le traitement des strates avec un seul élément
`one.stage`	si `TRUE`, ne traite qu'un seul niveau de sondage
`stage`	niveau de sondage en traitement
`cal`	contient les détails de la calibration.

La fonction `onestage()` est d'abord appliquée au premier niveau d'échantillonnage (ce qui correspond au calcul de A_h) puis, s'il y a un niveau d'échantillonnage supplémentaire, la fonction `multistage()` s'appelle elle-même pour chaque valeur différente des identificateurs d'unités d'échantillonnage, en adaptant les colonnes des différents paramètres. Les valeurs obtenues sont multipliées par un facteur correspondant au t_h qui multiplie B_h dans la formulation de $V_{2st,h}$ à la section 2.2.1. Toutes ces variances sont sommées et retournées.
Remarquons que la fonction `multistage()` peut gérer une calibration.

4.2.4 `svyrecvar()`

La fonction `svyrecvar()` gère la poststratification et appelle `multistage()`. Elle prend sept arguments

`x`	variable dont on calcule la variance
`clusters`	identificateurs des unités d'échantillonnage
`stratas`	identificateurs des strates
`fpcs`	tailles des populations totales
`postStrata`	détails concernant la poststratification
`lonely.psu`	définit le traitement des strates avec un seul élément
`one.stage`	si `TRUE`, ne traite qu'un seul niveau de sondage.

En premier lieu, le programme procède à la poststratification. La fonction `multistage()` est ensuite appelée. C'est à ce moment-là que l'option `"survey.lonely.psu"` est lue et passée en argument à `multistage()`.

4.3 Le calcul des statistiques

Nous détaillons à présent les fonctions `svytotal()`, `svyratio()`, `svymean()` et `svyquantile()` qui permettent, comme leurs noms l'indiquent, de calculer des totaux, des ratios, des moyennes et des quantiles. Elles peuvent traiter des données contenant plusieurs colonnes et donc calculer une statistique pour chacune d'elles. Toutefois, pour davantage de clarté, nous parlerons du total, du ratio et de la moyenne calculés, même s'il peut y en avoir plusieurs, suivant le format des données. Notons que pour le calcul de la variance, ces fonctions appellent `svyrecvar()`, que nous avons vu à la section 4.2.

4.3.1 `svytotal()`

La fonction `svytotal()` calcule un total pondéré. Elle prend quatre arguments

`x`	variable dont on calcule le total
`design`	design de l'enquête
`na.rm`	si `TRUE`, enlève les valeurs manquantes
`deff`	si `TRUE`, calcule l'effet du design.

Le paramètre `design` est censé être un objet de type `survey.design`, et doit donc avoir été calculé à l'aide de la fonction `svydesign()`. Les paramètres `na.rm` et `deff` ont `FALSE` comme valeur par défaut.
Pour commencer, le paramètre `x` est mis sous forme de matrice. Ensuite, si `na.rm = TRUE`, les valeurs manquantes sont ôtées des données. Enfin, le calcul du total est effectué grâce à la fonction `colSums()` qui calcule les sommes par colonnes de `x/design$prob`. C'est cette valeur qui est retournée par la fonction. La variance du total est calculée par la fonction `svyrecvar()`

```
svyrecvar(x/design$prob, design$cluster, design$strata, design$fpc,
postStrata=design$postStrata).
```

Pour terminer, l'effet du design est calculé. Pour cela, la variance empirique des données (obtenue grâce à `svyvar()`) est multipliée par

```
sum(weights(design)^2),
```

si `deff="replace"` et par

```
sum(w^2)*(sum(w)-nobs)/sum(w)
```

(où `w=weights(design)` et où `nobs` représente le nombre d'observations) si `deff=TRUE`. Le rapport entre la variance obtenue avec `svyrecvar()` et la valeur qui vient d'être calculée est alors renvoyé. Enfin, les différentes statistiques calculées sont retournées.
Précisons que c'est avec cette fonction qu'il a été possible de faire les calculs de variance pour la LSE, à l'aide des commandes

```
design  <-  svydesign(ids=~identr+NrSalaire, strata=~stragrs,
            data=data, weights=~gewibgrs, fpc=~th+thi)
med     <-  computeQuantiles(data$mbls, data$gewibgrs)
total   <-  svytotal(x=(data$mbls<=med)-0.5, design=design)
var     <-  attr(total, "var")/sum(data$gewibgrs)^2
se      <-  sqrt(var),
```

pour des données ne nécessitant pas d'imputation.

4.3.2 `svyratio()`

La fonction `svyratio()` calcule un ratio pondéré. Elle prend sept arguments

`numerator`	numérateur du ratio à estimer
`denominator`	dénominateur du ratio estimer
`design`	design de l'enquête
`separate`	si `TRUE`, traite chaque strate séparément
`na.rm`	si `TRUE`, enlève les valeurs manquantes
`formula`	alternative à `numerator`
`covmat`	si `TRUE`, calcule la matrice de covariance des ratios.

Les paramètres `separate`, `na.rm` et `covmat` sont par défaut fixés à `FALSE`. Si `separate=TRUE`, la fonction s'appelle elle-même pour chaque strate. Les paramètres `numerator` et `denominator` sont ensuite mis sous forme de matrice, après quoi, si `na.rm=TRUE`, les valeurs manquantes sont éliminées.
Les totaux pondérés de `numerator` et de `denominator` sont ensuite calculés grâce à la fonction `svytotal()` et le ratio est enfin effectué. C'est cette valeur qui est retournée par la fonction.
L'étape suivante est celle du calcul de la variance. La fonction `svyrecvar()` est appliquée à

$$\frac{\texttt{numerator-ratio*denominator}}{\sum\texttt{(denominator/design\$prob)}} * \frac{1}{\texttt{design\$prob}},$$

où `ratio` désigne le ratio qui vient d'être calculé. Les autres valeurs passées en arguments sont les paramètres habituels.
Pour terminer, si `covmat=TRUE`, la matrice de covariance des ratios est calculée.

4.3.3 svymean()

La fonction svymean() calcule une moyenne pondérée. Elle prend quatre arguments

x	variable dont on calcule la moyenne
design	design de l'enquête
na.rm	si TRUE, enlève les valeurs manquantes
deff	si TRUE, calcule l'effet du design.

Pour commencer, les données x sont mises sous forme de matrice, puis les valeurs manquantes sont éliminées si na.rm=TRUE. Enfin, la moyenne pondérée est calculée : la somme pondérée des données par design$prob est divisée par la somme des poids. C'est cette valeur qui est retournée. La fonction traite ensuite la variance : svyrecvar() est appliquée aux écarts à la moyenne pondérés

$$\frac{\texttt{sweep(x,2,average)}}{\texttt{design\$prob}} * \frac{1}{\sum \frac{1}{\texttt{design\$prob}}},$$

avec

$$\texttt{average <- colSums}\left(\texttt{x*}\frac{1}{\texttt{design\$prob}} * \frac{1}{\sum \frac{1}{\texttt{design\$prob}}}\right).$$

Les autres arguments de svyrecvar() sont les paramètres habituels. Pour terminer, si le paramètre deff a été spécifié, l'effet du design est établi. La variance empirique des données (à nouveau calculée par svyvar()) est cette fois-ci divisée par nobs dans le cas où deff="replace" et multipliée par

```
(sum(weights(design))-nobs)/(sum(weights(design))*nobs)
```

si deff=TRUE. C'est à nouveau le rapport entre la variance obtenue avec svyrecvar() et la valeur qui vient d'être calculée qui est renvoyé.

4.3.4 svyquantile()

La fonction svyquantile() calcule des quantiles pondérés. Elle prend huit paramètres

x	données dont il faut calculer les quantiles
design	design de l'enquête
quantiles	quantiles à calculer
ci	si TRUE, calcule des intervalles de confiance
alpha	1 - le niveau de confiance des intervalles
method	voir ci-dessous
f	voir ci-dessous
interval.type	voir ci-dessous.

Les paramètres alpha et ci ont 0.05 et FALSE comme valeurs par défaut. Les paramètres method (par défaut égal à "linear") et f (par défaut égal à 1) sont passés tels quels en argument lors de l'appel des fonctions approx() et approxfun() dans le corps de svyquantile(). Le paramètre interval.type détermine la méthode utilisée pour la construction de l'intervalle de confiance : "score" correspond à l'inversion d'un test du score robuste et "Wald" à l'inversion d'un intervalle de confiance construit sur l'échelle des pourcentages.
Le code de svyquantile() comprend trois parties. Dans la première, les données x sont mises sous forme de dataframe et les poids du design sont lus grâce à la fonction weights() (dont il est sujet à la section 4.4).

Dans la deuxième partie sont définies les fonctions `computeQuantiles()`, `computeScoreCI()` et `computeWaldCI()`. La fonction `computeQuantiles()`, comme son nom l'indique, calcule des quantiles pondérés. Les poids considérés sont ceux du design, alors que les données et les quantiles à calculer sont des paramètres de la fonction (`xx` et `p`). Les poids sont pris dans l'ordre croissant des données et leurs sommes partielles sont calculées puis divisées par la somme des poids totale. Une fonction de répartition empirique, `cdf`, est ensuite calculée à l'aide de la fonction `approxfun()`. Si `method="linear"`, `cdf` sera une approximation linéaire par morceaux et si `method="constant"`, `cdf` sera une fonction constante par morceaux (continue à droite si `f=0`, à gauche si `f=1`, voir l'aide de R au sujet de la fonction `approxfun()` pour davantage de détails). Enfin, `cdf(p)` est retourné.
C'est ensuite la fonction `computeScoreCI()` qui est définie. Elle permet le calcul d'un intervalle de confiance selon la méthode de l'inversion d'un test du score robuste qui est décrite dans [5]. Pour commencer, la fonction $\texttt{U}(\theta)$ est définie comme

$$\mathbb{1}_{\{\texttt{xx}>\theta\}}\texttt{-(1-p)}$$

et la fonction $\texttt{scoretest}(\theta,\ \texttt{qlimit})$, comme

$$\frac{\texttt{umean}}{\texttt{SE(umean)}}\texttt{-qlimit},$$

où $\texttt{umean=svymean(U(}\theta\texttt{),design)}$ et `SE(umean)` correspond à l'écart-type renvoyé par la fonction `svymean()`. La différence interquartile, `iqr`, est ensuite calculée grâce à la fonction `IQR()` (cette dernière ne permet pas de spécifier de poids, `iqr` est donc la différence des quartiles non pondérés). Cette valeur est utilisée pour la construction de l'intervalle

$$(\texttt{lower=min(xx)+iqr/100},\quad \texttt{upper=max(xx)-iqr/100}),$$

dans lequel la fonction `uniroot()` cherchera la racine de la fonction `scoretest()`, avec d'abord

$$\texttt{qlimit=qnorm(alpha/2, lower.tail=FALSE)}$$

puis

$$\texttt{qlimit=qnorm(alpha/2, lower.tail=TRUE)}.$$

Ce sont ces deux racines qui sont renvoyées comme bornes de l'intervalle de confiance.
Cette partie se termine avec la définition de la fonction `computeWaldCI()` qui inverse un intervalle de confiance construit sur l'échelle des pourcentages. Le quantile est d'abord calculé à l'aide de `computeQuantiles()`, puis la variable `U` est définie comme

$$\mathbb{1}_{\{\texttt{xx}>\theta_0\}}\texttt{-(1-p)}$$

et sa moyenne est calculée par `svymean()`. L'intervalle de confiance sur l'échelle des pourcentages est alors construit

```
p.up   <-  p + qnorm(alpha/2, lower.tail=FALSE)*SE(wtest)
p.low  <-  p + qnorm(alpha/2, lower.tail=TRUE)*SE(wtest),
```

où `p` est le quantile à calculer et `SE(wtest)`, l'écart-type renvoyé lors du calcul de la moyenne par `svymean()`. Les poids sont ensuite ordonnés selon l'ordre croissant des données, les sommes partielles calculées et divisées par la somme des poids totale. Enfin, la fonction `approx()` calcule les données correspondant à `p.up` et `p.low` suivant la méthode correspondant aux paramètres `method` et `f`. Les deux valeurs obtenues sont retournées.
Pour terminer, la troisième partie du code de `svyquantile()` voit le quantile, l'intervalle de confiance et l'écart-type calculés et retournés. Le quantile est d'abord obtenu à l'aide de

computeQuantiles(). Si ci=FALSE, ce quantile est retourné et le programme s'arrête. Sinon, la valeur spécifiée pour le paramètre interval.type est lue par l'intermédiaire de la forme à un argument de la fonction match.arg() (pour davantage de détails, voir l'aide de R au sujet de la fonction match.arg()). Après cela, la variable computeCI prend la valeur de computeScoreCI ou de computeWaldCI, suivant la valeur de interval.type ("score" ou "Wald"). L'intervalle de confiance est ensuite calculé par computeCI(). Enfin, l'écart-type est obtenu par la division de la longueur de l'intervalle de confiance par

2*qnorm(alpha/2, lower.tail=FALSE). (1)

Le quantile, l'intervalle de confiance et l'écart-type sont enfin retournés.

4.4 Autres fonctions

Les fonctions suivantes font également partie du package *survey* et peuvent se révéler utiles lors du calcul d'un intervalle de confiance pour les données d'une enquête avec un design complexe.

La fonction update() permet de mettre à jour un design en y ajoutant une variable (son code est accessible via la commande getS3method("update", "survey.design")).

La fonction subset() permet de restreindre le design à une sous-population et ainsi la variable manipulée sera moins volumineuse (de manière similaire, son code est accessible via getS3method("subset", "survey.design")).

La fonction weights() a deux formes spécifiques au package survey. La première permet d'extraire les poids d'un design, pour cela il suffit de l'appliquer à un objet de classe survey.design. La deuxième, elle, calcule des poids d'extrapolation à l'aide d'un objet de type fpc. Cette dernière est d'ailleurs appelée dans la fonction svydesign() si ni probs, ni weights n'ont été spécifiés. Les codes de ces deux fonctions sont accessibles via getS3method("weights", "survey.design") et getS3method("weights", "survey_fpc") (le code de la fonction s'appliquant à un objet de type fpc est également disponible dans le fichier Multistage.R dont il est sujet au début du chapitre 4).

La fonction svyvar() effectue un calcul de variance empirique. Elle appelle svymean() qui calcule d'abord la moyenne pondérée des données, xbar, puis la moyenne pondérée de

(x-xbar)^2(n/(n-1)),

où n est le nombre d'observations. Le code de svyvar() est accessible via la commande

getS3method("svyvar","survey.design").

Enfin, nous avons trouvé à l'adresse http://www.dcs.napier.ac.uk/peas/R/myRfunctions.R le code d'une alternative à la fonction svyquantile(), my.svyquantile(). Les données y sont d'abord mises sous une forme adéquate, puis une fonction computeQuantiles() est définie. Cette dernière appelle approxfun(), avec le paramètre method=linear et est tout de suite utilisée pour le calcul du quantile. La fonction getpse() est ensuite définie, dans laquelle le design est mis à jour par l'ajout de la variable pct=1*(x<rv), où x désigne les données et rv le paramètre passé en argument à la fonction getpse() (ce sera la variable Quantile, qui contient le quantile calculé plus haut), puis un écart-type est calculé à partir de la variance renvoyée par svymean(pct, design). La variable sep stocke l'écart-type obtenu en appliquant getpse() à Quantile. Cet écart-type permet le calcul d'un intervalle de confiance à 68.62%, qui, divisé par 2, donne l'estimation de l'écart-type qui sera retournée. Enfin, toujours à partir de sep, un intervalle de confiance à 95% est calculé (avec qnorm). Finalement, les valeurs de quantiles (paramètre), Quantile (calculé par computeQuantiles()), se (intervalle

de confiance à 68.62% divisé par 2), `l.limit` et `u.limit` (intervalle de confiance à 95%) sont renvoyées. Pour résumer, on a un quantile calculé à l'aide de la fonction `approxfun()` avec le paramètre `method="linear"`, un écart-type calculé par division d'un intervalle de confiance à 68.62% par deux (cet intervalle de confiance ayant lui été construit par la fonction `approxfun()` avec le paramètre `method="linear"` avec un écart-type calculé par `svymean()`) et un intervalle de confiance à 95% construit lui aussi par `approxfun()` (toujours avec le paramètre `method="linear"`) avec `qnorm` et un écart-type calculé par `svymean()`.

4.5 Application à la LSE

Comme nous l'avons vu, la méthode de calcul de la variance appliquée dans le cadre de la LSE correspond à ce qui a été implémenté dans le package *survey* (excepté pour ce qui est de l'imputation au niveau des strates). Cependant, c'est grâce à `svytotal()` qu'il est possible d'effectuer le calcul de variance plutôt qu'avec `svyquantile()`, `svyratio()` ou encore `svymean()`, même s'il s'agit du calcul de la variance d'une médiane et qu'il se rapproche davantage du calcul d'un ratio que de celui d'un total. Nous allons détailler les raisons pour lesquelles `svytotal()` est la seule fonction permettant de reproduire les calculs de la LSE.
La première variable à passer en argument à `svyrecvar()` pour obtenir la bonne variance est

$$\texttt{data\$gewibgrs*((data\$mbls<=med)-0.5)}.$$

La fonction `svytotal()` passe `x/design$prob` en premier argument lors de l'appel de la fonction `svyrecvar()`. Si on spécifie `x=(data$gewibgrs<=med)-0.5` lors de l'appel de `svytotal()`, la variance retournée sera la bonne (il faudra encore la diviser par le carré de la somme des poids totale, voir la section 4.3.1). Dans ce cas, le calcul de la médiane doit tout de même être effectué à part.
Dans le corps de la fonction `svyratio()`, `svyrecvar()` est appelé avec

$$\frac{\texttt{numerator-ratio*denominator}}{\texttt{sum(denominator/design\$prob)}} * \frac{\texttt{1}}{\texttt{design\$prob}}$$

comme premier argument. Pour obtenir la bonne variance, il faudrait spécifier

$$\texttt{numerator=(data\$mbls<=med)} \quad \text{et} \quad \texttt{denominator=1}.$$

Il faudrait encore que `ratio=0.5`, mais `0.5` n'est qu'une valeur théorique et `ratio` sera en général différent (rappelons que `ratio` est le quotient de `svytotal()` appliqué à `numerator` et à `denominator`). Enfin, il reste la division par

$$\texttt{sum(denominator/design\$prob)}$$

qui pose également problème.
Dans le cas de `svymean()`, `svyrecvar()` est appelé avec

$$\frac{\texttt{sweep(x,2,average)}}{\texttt{design\$prob}} * \frac{\texttt{1}}{\texttt{sum(1/design\$prob)}}$$

comme premier argument, où

$$\texttt{average = colSums}\left(\frac{\texttt{x}}{\texttt{design\$prob}} * \frac{\texttt{1}}{\texttt{sum(1/design\$prob)}}\right).$$

Pour que le résultat soit celui que l'on attend, il faudrait poser

$$\texttt{x = (data\$mbls<=med)} \quad \text{et} \quad \texttt{average = 0.5}.$$

Il y a de nouveau la différence entre la valeur théorique (`0.5`) et la valeur calculée (`average`). De plus, il y a le facteur

$$\frac{1}{\texttt{sum(1/design\$prob)}}$$

qui ne correspond à rien.

Enfin, pour terminer, voici les différences entre les calculs effectués dans le cadre de la LSE et ce que la fonction `svyquantile()` permet de faire. Lors du calcul des données correspondant à un pourcentage (un quantile ou une borne d'intervalle), il y a déjà une différence, même si l'on spécifie `method="constant"` et `f=1`. L'exemple suivant en est une illustration. Considérons le vecteur `x <- 1:10` et les poids associés `w <- rep(.1,10)`. Selon la méthode de la LSE, la médiane vaudra `5.5` (voir la description de la méthode de la LSE, section 2.1), alors que la fonction `svyquantile()` renverra `5`, car elle procède à une interpolation autour, entre autres, du point (5, 0.5).

Au niveau de l'intervalle de confiance (en spécifiant `interval.type="Wald"`) il y a plusieurs différences. La fonction `svyquantile()` utilise l'écart-type renvoyé par `svymean()` (qui est donc l'écart-type des écarts à la moyenne) et le résultat de `qnorm()` pour la construction de l'intervalle de confiance sur l'échelle des pourcentages, alors que selon la méthode de la LSE, l'écart-type du percentile est calculé de manière différente puis multiplié par 1.96. De plus, l'inversion de l'intervalle de confiance ne se fait pas de la même façon. Enfin, l'écart-type renvoyé par `svyquantile()` est obtenu en divisant l'intervalle de confiance à 95% par 2, alors que dans le cadre de la LSE, pour le calcul de CV_{perc}, on utilise l'écart-type du percentile et pour le calcul des coefficients de variation synthétiques, on utilise le plus grand demi-intervalle de confiance.

4.5.1 Résultats

Dans cette section nous allons d'abord présenter les résultats obtenus avec `lseComp()` et avec `svyquantile()` (pour tous les paramètres `method` et `f` possibles) pour les classes NOGA 2 10 et 40 de la grande région 4.

Tableau 1 Résultats pour la classe NOGA 2 10 de la grande région 4.

	med	bi	bs
`lseComp()`	5740	5616	5977
`"Wald", "constant", 0`	5739	5608	5960
`"Wald", "constant", .5`	5739.5	5608	5968.5
`"Wald", "constant", 1`	5740	5616	5977
`"Wald", "linear"`	5739.815	5608	5965.985
`"score", "constant", 0`	5739	5591	5960
`"score", "constant", .5`	5739.5	5591	5960
`"score", "constant", 1`	5740	5591	5960
`"score", "linear"`	5739.815	5591	5960

Comme on pouvait s'y attendre, on voit sur les tableaux 1 et 2 que ce sont les valeurs

`interval.type="Wald",` `method="constant"` et `f=1`

qui donnent les résultats les plus proches de ceux de `lseComp()`. Dans le tableau 3, on trouve les résultats de `lseComp()` et de `svyquantile()` avec les paramètres `interval.type="Wald"`, `method="constant"` et `f=1` pour chaque classe NOGA 2 de la grandes région 4 (`nog` désigne la classe NOGA 2, `ml`, `bil` et `bsl`, la médiane et les bornes de l'intervalle de confiance calculées

Tableau 2 Résultats pour la classe NOGA 2 40 de la grande région 4.

	med	bi	bs
lseComp()	7047	6973	7158
"Wald", "constant", 0	7037	6929	7154
"Wald", "constant", .5	7042	6951	7156
"Wald", "constant", 1	7047	6973	7158
"Wald", "linear"	7037.124	6951.948	7157.108
"score", "constant", 0	7037	6985	7158
"score", "constant", .5	7042	6985	7158
"score", "constant", 1	7047	6985	7158
"score", "linear"	7037.124	6985	7158

par `lseComp()`, `ms`, `bis` et `bss`, la médiane et les bornes de l'intervalle de confiance calculées par `svyquantile()`).
On remarque que les résultats correspondent souvent. Notons que ce n'est qu'avec la version 3.6-6 du package *survey* qu'il a été possible d'approcher autant les résultats de la LSE avec la fonction `svyquantile()`. En effet, dans les versions précédentes, le code de la fonction `svyquantile()` comportait un bug qui ne permettait pas le calcul avec `method="constant"`.
Ajoutons enfin que la fonction `svyquantile()` fournit bien un quantile, un intervalle de confiance à 95% et un écart-type, mais pas d'intervalle de confiance à 68.62%, ni aucun coefficient de variation.
Les calculs ont également été effectués (pour les classes NOGA 2 de la grande région 4) avec la fonction `lseComp()`, en remplaçant `1.96` par `qnorm()`. Les résultats obtenus sont identiques à ceux retournés par la fonction `lseComp()` originale.

Tableau 3 Comparaison entre `lseComp()` et `svyquantile()` (avec `interval.type="Wald"`, `method="constant"` et `f=1`) pour quelques classes NOGA 2 de la grande région 4. `ml` et `ms` représentent les médianes, `bil` et `bis`, les bornes inférieures de l'intervalle de confiance et `bsl` et `bss`, les bornes supérieures.

	lseComp()			svyquantile()		
nog	ml	bil	bsl	ms	bis	bss
10	5740	5616	5977	5740	5616	5977
15	5312	5093	*5497*	5312	5093	*5498*
17	4588	4487	4720	4588	4487	4720
18	4533	4282	4767	4533	4282	4767
19	8942	*7415*	*10060*	8942	*7054*	*10569*
20	5383	5232	5575	5383	5232	5575
21	5551	5305	5735	5551	5305	5735
22	6718	*6591*	6890	6718	*6589*	6890
23	6382	6252	6476	6382	6252	6476
25	5285	5187	5439	5285	5187	5439
26	5596	5480	5712	5596	5480	5712
27	5674	5587	5778	5674	5587	5778
29	6608	*6527*	6695	6608	*6526*	6695
30	7341	*6977*	7677	7341	*6976*	7677
33	6338	6226	6482	6338	6226	6482
36	5496	5396	5582	5496	5396	5582
40	7047	6973	7158	7047	6973	7158
45	5666	5598	5744	5666	5598	5744

5 Performances

Nous avons mesuré, à l'aide de la fonction `system.time()`, les temps que mettent le programme et le package (plus précisément, `svydesign()` et `svytotal()`) pour traiter différentes classes de salaires. Pour l'utilisation du package, l'ajout de la colonne `NrSalaire`, ainsi que le calcul de la médiane par `computeQuantiles()` n'ont pas été pris en compte.
Considérons, pour commencer, le temps de lecture des données. Le fichier (de type .csv) pour la LSE02 contient 1'031'538 salaires. La commande

```
data <- read.table("pathname",sep=";",header=TRUE)
```

met 530.18 secondes a être effectuée.
Considérons un nombre de salaires relativement petit. La classe NOGA 10 de la grande région 4 en compte 300. La spécification du design par la commande

```
design  <-  svydesign(ids=~identr+NrSalaire, strata= stragrs,
            data=data, weights=~gewibgrs, fpc=~th+thi)
```

prend 0.08 secondes et le calcul de la variance par la fonction `svytotal()`

```
total <- svytotal(x=(data$mbls<=med)-0.5, design=design)
```

dure 0.12 secondes. La fonction `lseComp()` met, elle, 0.09 secondes pour traiter ce cas.
Passons à présent au secteur secondaire de la grande région 4, ce qui représente 57'266 salaires. La fonction `svydesign()` a besoin de 613.14 secondes pour en déterminer le design.

`svytotal()`, sans effectuer d'imputation, traite ces salaires en 105.11 secondes, alors qu'il suffit de 1.55 secondes à `lseComp()`.
En ce qui concerne la grande région 4 en entier, il s'agit là de 230'443 salaires, le package atteint ses limites. En effet, lors de l'exécution de la fonction `svydesign()`, un message d'erreur apparaît sans que le design n'ait pu être spécifié

```
Erreur : impossible d'allouer un vecteur de taille 900 Ko
Messages d'avis : Reached total allocation of 765Mb.
```

La fonction `lseComp()`, par contre, met 7.40 secondes pour traiter ces salaires. Enfin. pour les 1'031'538 salaires, `lseComp()` a besoin de 61.38 secondes.
Notons que seules les valeurs relatives de ces temps sont importantes, car ils dépendent de l'ordinateur sur lequel les calculs ont été effectués.

Tableau 4 Performances du programme et du package *survey* pour quelques classes de salaires.

	gr4 nog10	gr4 sect. sec.	gr4	CH
salaires	300	57'266	230'443	1'031'538
`svydesign()`	0.08	613.14	impossible	impossible
`svytotal()`	0.12	105.11	impossible	impossible
`lseComp()`	0.09	1.55	7.40	61.38

Conclusion

Le but poursuivi au début de ce travail était de déterminer s'il est possible d'utiliser le package *survey* pour les calculs d'intervalles de confiance de la médiane dans le cadre de la LSE. Nous sommes arrivés à la conclusion que *survey* ne constitue pas un outil adéquat, pour deux raisons. Premièrement, certaines fonctions du package (`svydesign()`, par exemple, qui est pourtant indispensable à l'utilisation du package) n'arrivent pas à gérer des données d'une taille aussi importante (voir chapitre 5, le fichier total pour la LSE02 est constitué de plus d'un milion de salaires et occupe plus de 70 Mo ; la grande région 4 compte 230'443 salaires et là déjà la fonction `svydesign()` est dépassée). Deuxièmement, les méthodes implémentées ne correspondent pas exactement aux calculs effectués dans le cadre de la LSE, comme par exemple dans le cas de la médiane (même avec le paramètre `method="constant"`, la médiane renvoyée par la `svyquantile()` ne correspond pas à celle calculée par la méthode de la LSE). L'étude du package *survey* a néanmoins motivé l'implémentation d'un programme spécifique et a permis la description d'une partie du package (laquelle description constitue une grande partie du présent rapport). Le programme, lui, offre des performances raisonnables et constitue donc un outil utilisable dans la pratique.

Annexes

A Mode d'emploi

lseComp — Médiane pour la LSE

Description

Calcule une moyenne pondérée ainsi qu'un intervalle de confiance et plusieurs coefficients de variation pour les salaires de la LSE.
x

Usage

```
lseComp(data, ..., noga_spec=NULL)
```

Arguments

`data`	Data frame contenant une colonne pour chacun des éléments suivants : `mbls`, `stragrs`, `identr`, `nrep`, `th`, `anzlohn`, `thi`, `gewibgrs`, `gr` et `nog_2` (voir Détails pour plus d'informations).
`...`	Données selon lesquelles les résultats seront détaillés.
`noga_spec`	Vecteur spécifiant les groupes de classes noga pour lesquelles la médiane doit être calculée.

Détails

La fonction `lseComp` appelle la fonction `statMed` qui calcule la médiane pondérée (à l'aide de la fonction `computeQuantiles`), l'intervalle de confiance à 95%, les coefficients de variation synthétiques à 95% et à 68.62% et le coefficient de variation du percentile. `mbls` désigne les salaires, `stragrs` les strates, `identr` les identificateurs d'entreprises, `nrep` le nombre d'entreprises ayant fourni au moins un salaire, `th` les taux de sondage effectifs dans les strates, `anzlohn` le nombre de salaires fournis par entreprises, `thi` les taux de sondage effectifs dans les entreprises, `gewibgrs` le produit des taux d'occupation et des poids d'extrapolation, `gr` les régions NUTS II et `nog_2` les regroupements NOGA2 utilisés dans la LSE.

Valeur

Renvoie un data frame.

Remarque

Il est possible d'obtenir la médiane pondérée d'un domaine en appliquant directement `lseComp` au sous-ensemble de données correspondant. Toutefois, les méthodes de calcul reposant sur des principes de convergence asymptotiques, les résultats obtenus pour des domaines de taille restreinte sont à interpréter avec prudence.

Références

Graf, M. (2002). Enquête suisse sur la structure des salaires 2000. Plan d'échantillonnage, pondération et méthode d'estimation pour le secteur privé. *Rapport de méthode 338-0010*, Office fédéral de la statistique, http://www.bfs.admin.ch/bfs/portal/fr/index/infothek/erhebungen__quellen/methodenberichte.Document.50660.pdf.

Exemples

Soit `data` le fichier contenant l'ensemble des données. La commande suivante calcule la médiane pondérée de l'ensemble des données

```
lseComp(data).
```

Afin d'effectuer le calcul uniquement pour la région NUTS II 4, on utilisera

```
lseComp(data[data$gr==4,])
```

et pour la classe noga 10-14 de la région NUTS II 4 (codée par 10 dans le jeu de données),

```
lseComp(data[data$gr==4 & data$nog_2==10,]).
```

Pour traiter l'ensemble des données, mais avec la médiane calculée pour chaque classe noga séparément

```
lseComp(data, ~nog_2).
```

Dans les exemples suivants, `GESCHLE` est un nom de colonne qui spécifie le sexe des employés. Il pourrait s'agir de n'importe quel nom de colonne pour laquelle des résultats détaillés sont voulus. La commande suivante effectue le calcul pour l'ensemble des données, mais avec la médiane calculée pour chaque sexe séparément

```
lseComp(data, ~GESCHLE).
```

Pour l'ensemble des données, mais avec la médiane calculée pour chaque classe noga et chaque sexe séparément

```
lseComp(data, ~nog_2, ~GESCHLE).
```

Il est possible de mélanger des classes noga et des groupes de classes noga

```
lseComp(data, noga_spec=c(10, 15:37, list(15:37))).
```

Pour obtenir les résultats pour l'ensemble des données, pour chaque classe noga et pour quelques groupes de classes noga

```
lseComp(data, noga_spec=c(list(unique(data$nog_2)), 1, list(10:45),
list(10:14), list(15:37), 15:45, list(50:93), list(50:52), 50:55,
list(60:64), 60:64, list(65:67), 65:67, list(70:74), 70:85,list(90:93),
90:93)).
```

Idem, mais pour chaque sexe séparément

```
lseComp(data, ~GESCHLE, noga_spec = c(list(unique(data$nog_2)), 1,
list(10:45), list(10:14), list(15:37), 15:45, list(50:93), list(50:52),
50:55, list(60:64), 60:64, list(65:67), 65:67, list(70:74), 70:85,
list(90:93), 90:93)).
```

Description

Calcule une médiane pondérée avec un intervalle de confiance et plusieurs coefficients de variation pour des données issues d'un plan de sondage complexe, pour un domaine quelconque et un schéma de poids général.

Usage

```
statMed(x, strata=NULL, psu=NULL, nh=NULL, th=NULL, Nh=NULL, mhi=NULL, thi=NULL,
Mhi=NULL, weights=NULL, crit=NULL)
```

Arguments

`x`	Variable d'étude dont la médiane est calculée
`strata`	Identificateurs des strates
`psu`	Identificateurs des PSU
`nh`	Tailles d'échantillonnage nettes des strates (dans l'échantillon original, indépendamment de toute définition de domaine)
`Nh`	Tailles totales des strates
`th`	Taux d'échantillonnage nets des strates (`nh/Nh`)
`mhi`	Tailles d'échantillonnage des PSU (dans l'échantillon original, indépendamment de toute définition de domaine)
`Mhi`	Tailles totales des PSU
`thi`	Taux d'échantillonnage des PSU (`mhi/Mhi`)
`weights`	Poids des SSU (tenant compte, mais pas nécessairememt égaux aux poids d'échantillonnage)
`crit`	Critère utilisé pour l'imputation de la variance pour les strates ne contenant qu'un seul PSU

Détails

Pour commencer, la médiane pondérée est calculée par `computeQuantiles`. Puis, la variance "intra-PSU" est calculée (posée égale à 0 si le taux de sondage vaut un). La variance "inter-PSU" est également calculée, avec une imputation pour les strates ne contenant qu'un seul PSU, basée sur les strates appartenant au même critère "crit" (l'imputation n'a lieu que s'il n'y a qu'un seul PSU dans la strate). Pour terminer, un intervalle de confiance à 95%, des coefficients de variation synthétiques à 95% et à 68.62% et le coefficient de variation du percentile sont calculés.

Valeur

Renvoie un data frame.

Remarque

Il est possible d'obtenir la médiane pondérée d'un domaine en appliquant directement `lseComp` au sous-ensemble de données correspondant. Toutefois, les méthodes de calcul reposant sur des principes de convergence asymptotiques, les résultats obtenus pour des domaines de taille restreinte sont à interpréter avec prudence.

Références

Graf, M. (2002). Enquête suisse sur la structure des salaires 2000. Plan d'échantillonnage, pondération et méthode d'estimation pour le secteur privé. *Rapport de méthode 338-0010*, Office fédéral de la statistique, http://www.bfs.admin.ch/bfs/portal/fr/index/infothek/erhebungen__quellen/methodenberichte.Document.50660.pdf.

Exemple

Voici la spécification à utiliser dans le cadre de la LSE (Enquête suisse sur la structure des salaires). Soit `data` un data frame contenant toutes les données : `mbls` étant la variable dont la médiane doit être établie, `stragrs`, `identr`, etc, étant les données correspondant aux strates, aux PSU, etc. Finalement, supposons que l'imputation doit se faire selon les valeurs de `gr` et `nog_2`.

```
critgrnog_2 <- as.numeric(paste(data$gr, data$nog_2, sep=""))
statMed(x=data$mbls, strata=data$stragrs, psu=data$identr, nh=data$nrep,
th=data$th, mhi=data$anzlohn, thi=data$thi, weights=data$gewibgrs,
crit=critgrnog_2)
```

B Code R

Voici pour commencer un script qui pourra être utilisé pour mettre les données fournies par la section LOHN de l'OFS sous une forme compatible avec les programmes.

```
#Il faut d'abord convertir avec Excel les deux fichiers .xls en .csv
#(ziehungsplan et struktur), puis les lire.
ziehung <- read.table(file="pathname", sep=";", header=TRUE)
struktur <- read.table(file="pathname", sep=";", header=TRUE)

#Calculer ensuite les nombres à passer en arguments à la fonction read.fwf.
struktur$Variablen_Bereich <- as.numeric(substring(struktur$Variablen_Bereich,
    7,9)) - as.numeric(substring(struktur$Variablen_Bereich, 1,3)) + 1
#Effacer la dernière ligne ("NA").
struktur <- struktur[!is.na(struktur$Variablen_Bereich),]

#Spécifier les variables conservées.
usefulvariables <- c("BURNR_N", "GESCHLE", "ANFORNI", "INITGEW2", "MBLS",
    "GEWIBGRS", "NOG_2", "ANZLOHN", "GR", "STRA_N")
struktur$Variablen_Bereich[!'%in%'(struktur$Variablen_Name,usefulvariables)] <-
    - struktur$Variablen_Bereich[!'%in%'(struktur$Variablen_Name,
    usefulvariables)]

#Lire les données.
data <- read.fwf(file="pathname",widths=struktur$Variablen_Bereich,
    col.names=struktur$Variablen_Name[struktur$Variablen_Bereich>0],
    buffersize=1000)

#Ne conserver que les observations avec GESCHLE et ANFORNI renseigné.
data <- data[data$ANFORNI>0,]
data <- data[data$GESCHLE>0,]
rm(struktur)

#Ajouter les données de ziehung à data.
data <- merge(data, ziehung, by.x="STRA_N", by.y="stra_n")
rm(ziehung)

#Calculer les taux th et thi.
data$thi <- 1/data$INITGEW2
data <- data[,names(data)!="INITGEW2"]
data$th <- data$nrep/data$burnh
data <- data[,names(data)!="burnh"]

#Renommer les colonnes conformément aux programmes.
names(data)[names(data)=="BURNR_N"] <- "identr"
names(data)[names(data)=="MBLS"] <- "mbls"
names(data)[names(data)=="GEWIBGRS"] <- "gewibgrs"
names(data)[names(data)=="NOG_2"] <- "nog_2"
names(data)[names(data)=="ANZLOHN"] <- "anzlohn"
names(data)[names(data)=="GR"] <- "gr"
names(data)[names(data)=="STRA_N"] <- "stragrs"
```

```
#Charger le code des programmes (CImedLSE.R) et exécuter lseComp.
source("pathname")
lseComp(data)
```

Et voici à présent le code des programmes proprement dit.

```
#returns the median, the 95% confidence interval and three
#coefficients of variation

lseComp <- function(data,..., noga_spec=NULL) {

    Nssu <- dim(data)[1]
    if (Nssu>0) {

        det <- list(...)

        #if a column name has been specified, makes the computation
        #for each value of this column
        if (length(det)>0) {
            if (length(det)==1) {
                ret <- lseComp(data, noga_spec=noga_spec)
                det <- det[[1]]
                det <- model.frame(det,data=data)[,1]
                for (i in unique(det))
                    ret <- rbind(ret, lseComp(data[det == i,],
                             noga_spec=noga_spec))
                return(ret)
            } else {
                #more than one column have been specified
                ret <- lseComp(data, det[[-1]], noga_spec=noga_spec)
                detc <- det[[1]]
                detc <- model.frame(detc,data=data)[,1]
                for (i in unique(detc))
                    ret <- rbind(ret, lseComp(data[detc == i,], det[[-1]],
                noga_spec=noga_spec))
                return(ret)
            }
        }

        #if a noga specification has been mentionned,
        #calls lseComp for each group of noga class (a group
        #may contain only one class)
        if (!is.null(noga_spec)) {
            ret <- NULL
            for (i in 1:length(noga_spec))
                ret <- rbind(ret, lseComp(data[data$nog_2 %in%
                noga_spec[[i]],], noga_spec=NULL))
            return(ret)
        }
```

```
        #if none of the precedent conditions is TRUE, applies
        #statMed to the whole dataset
        gr <- unique(data$gr)
        gr <- ifelse(length(gr)==1,
                as.character(gr),
                paste(as.character(min(gr)), "-",
                  as.character(max(gr)), sep=""))
        noga <- unique(data$nog_2)
        noga <- ifelse(length(noga)==1,
                as.character(noga),
                paste(as.character(min(noga)), "-",
                  as.character(max(noga)), sep=""))
        GESCHLE <- unique(data$GESCHLE)
        GESCHLE <- ifelse(length(GESCHLE)==1,
                as.character(GESCHLE),
                paste(as.character(min(GESCHLE)), "-",
                as.character(max(GESCHLE)), sep=""))
        ANFORNI <- unique(data$ANFORNI)
        ANFORNI <- ifelse(length(ANFORNI)==1,
                as.character(ANFORNI),
                paste(as.character(min(ANFORNI)), "-",
                as.character(max(ANFORNI)), sep=""))
        ta3 <- unique(data$ta3)
        ta3 <- ifelse(length(ta3)==1,
                as.character(ta3),
                paste(as.character(min(ta3)), "-",
                  as.character(max(ta3)), sep=""))
        #computes the imputation criteria
        critgrnog_2 <- as.numeric(paste(data$gr, data$nog_2,
                         sep=""))
        ret <- data.frame(gr=gr, noga=noga, GESCHLE=GESCHLE,
                 ANFORNI=ANFORNI, ta3=ta3, statMed(x=data$mbls,
                 strata=data$stragrs, psu=data$identr, nh=data$nrep,
                 th=data$th, mhi=data$anzlohn, thi=data$thi,
                 weights=data$gewibgrs, crit=critgrnog_2))
        return(ret)
    }
}


#calculates a weighted median, its variance, a 95% confidence
#interval and three coefficients of variation

statMed <- function(x, strata=NULL, psu=NULL, nh=NULL, th=NULL,
             Nh=NULL, mhi=NULL, thi=NULL, Mhi=NULL,
             weights=NULL, crit=NULL) {

    #definition of a counting function
    compt <- function(variable) {return(length(unique(variable)))}
```

```
#if no stratum (or no psu), assumes there is only one stratum
#(or psu)
if (is.null(strata))
    strata <- rep(1, length(x))
if (is.null(psu))
    psu <- rep(1, length(x))
stra <- tapply(strata, psu, unique)

#checks the data: PSU level
ifelse (is.null(nh),
    #assumes domain corresponds to strata
    nh <- tapply(psu, strata, compt),
    nh <- tapply(nh, strata, unique)
)
ifelse (!is.null(Nh),
    {ifelse (!is.null(th),
        {Nh <- tapply(Nh, strata, unique)
         th <- tapply(th, strata, unique)},
        {Nh <- tapply(Nh, strata, unique)
         th <- nh/Nh}
    )},
    {ifelse (!is.null(th),
        {th <- tapply(th, strata, unique)
         Nh <- nh/th},
        #assumes total sampling
        {th <- rep(1, compt(stra))
         Nh <- nh}
    )}
)

#checks the data: SSU level
ifelse (is.null(mhi),
    #assumes domain corresponds to strata
    mhi <- tapply(x, psu, compt),
    mhi <- tapply(mhi, psu, unique)
)
ifelse (!is.null(Mhi),
    {ifelse (!is.null(thi),
        {Mhi <- tapply(Mhi, psu, unique)
         thi <- tapply(thi, psu, unique)},
        {Mhi <- tapply(Mhi, psu, unique)
         thi <- mhi/Mhi}
    )},
    {ifelse (!is.null(thi),
        {thi <- tapply(thi, psu, unique)
         Mhi <- mhi/thi},
        #assumes total sampling
        {thi <- rep(1, compt(psu))
         Mhi <- mhi}
    )}
```

```
)

#checks the data: weigths
if (is.null(weights))
    #computes sampling weights
    weights <- 1/(th*thi)

#computation of the median
med <- computeQuantiles(x, weights, .5)

#for each SSU:
zhij <- 1 * (x <= med)
ej <- weights*(zhij-.5)

#computation of the intra-PSU variance for each PSU:
ehi <- tapply(ej, psu, sum)
Bhi <- tapply(ej, psu, var)
#number of SSU in the domain
NDhi <- tapply(rep(1, length(x)), psu, sum)
Bhi <- ((NDhi-1)*Bhi+NDhi*(1-NDhi/mhi)*(ehi/NDhi)^2)*
         (1-thi)*mhi/(mhi-1)
#avoids NA if thi=1
Bhi[thi==1] <- 0
#sum of the weights in the PSU
svhi <- tapply(weights, psu, sum)

#sum of the intra-PSU variances
#if there is only one psu in the stratum, and only one Bhi, returns
#this Bhi; otherwise, returns the sum of the Bhi in the stratum
#which are different from NA
Bh <- tapply(Bhi, stra, function(v) {
    if (length(v)==1) return(v)
    if (length(v)>1) return(sum(v[!is.na(v)]))
})

#computation of the inter-PSU variances for each stratum
eh <- tapply(ehi, stra, sum)
Ah <- tapply(ehi, stra, var)
toth <- tapply(NDhi, stra, sum)
dlh <- tapply(NDhi-1, stra, sum)
#number of SSU in the domain
ne <- toth - dlh
Ah[ne>1] <- ((ne[ne>1]-1)*Ah[ne>1] + ne[ne>1]*(1-ne[ne>1]/nh[ne>1])
              *(eh[ne>1]/ne[ne>1])^2) / (nh[ne>1]-1)
#sum of the weights in the stratum
svh <- tapply(svhi, stra, sum)

#checks if the imputation has to be done and, if it is necessary,
#does it
if (!is.null(crit) & any(is.na(Ah)) & length(Ah[Ah!=NA])>0) {
```

```
        #computes a relative Ah variance
        Ahrel <- Ah/svh^2
        crit <- tapply(crit, strata, unique)
        #takes the mean of the relative variances for each value of crit
        m_Ahrel <- tapply(Ahrel[!is.na(Ahrel)], crit[!is.na(Ahrel)], mean)
        #proceeds to the imputation
        for (i in 1:length(Ah[ne==1])) {
            lab <- names(Ah[ne==1][i])
            lab <- crit[names(crit)==lab]
            #imputes the computed value if it exists
            if (lab %in% names(m_Ahrel))
                Ah[ne==1][i] <- svh[ne==1][i]^2*
                                  m_Ahrel[names(m_Ahrel)==lab]
        }
    }

    #avoids NA if th=1
    Ah[ne==1 & abs(th-1)<0.0000001] <- 0
    Ah <- Ah*nh*(1-th)

    #final computations
    #computes the variances of the strata
    V2sth <- ifelse (abs(th-1)<0.0000001, Bh, Ah+th*Bh)
    #total sum of weights
    denom <- sum(svh)
    #the global variance
    SV2st <- sum(V2sth[!is.na(V2sth)])/denom^2
    #the standard error
    sep <- sqrt(SV2st)
    #the 95% confidence interval
    l.limit <- computeQuantiles(x, weights, 0.5 - 1.96 * sep)
    u.limit <- computeQuantiles(x, weights, 0.5 + 1.96 * sep)
    #the three coefficients of variation
    cv_s95 <- 100*max(med-l.limit, u.limit-med)/(1.96*med)
    cl <- computeQuantiles(x, weights, 0.5-sep)
    cu <- computeQuantiles(x, weights, 0.5+sep)
    cv_s <- 100*max(med-cl, cu-med)/med
    CVperc <- 100*sep/0.5
    #the number of strata, PSU and SSU
    Nstrata <- compt(strata)
    Npsu <- compt(psu)
    Nssu <- length(x)

    return(cbind(l.limit, u.limit, median=med, cv_s95, cv_s,
      CVperc, Nstrata, Npsu, Nssu))
}
```

```
#calculates weighted quantiles

computeQuantiles <- function(xx, ww, qq = 0.5){

    #if no weights have been specified, returns non
    #weighted quantiles
    if (missing(ww))
        return(quantile(xx,probs=qq,na.rm=T))

    #otherwise, computes the partial sums of ww
    ord <- order(xx)
    cum.w <- cumsum(ww[ord])[!is.na(xx)]/sum(ww[!is.na(xx)])
    tmpS <- data.frame(matrix(rep(NA,2*length(qq)),nrow=2))
    tmpO <- data.frame(matrix(rep(NA,2*length(qq)),nrow=2))
    res <- c(rep(NA, length(qq)))

    #and computes each quantile
    for (i in 1:length(qq)) {
        #records the two sums directly greater than qq
        tmpS[i] <- cum.w[cum.w>=qq[i]][1:2]
        #and the corresponding orders (*)
        tmpO[i] <- ord[cum.w>=qq[i]][1:2]
        #if a sum is equal to qq, returns the mean of the two xx
        #corresponding to (*), otherwise, the lowest
        res[i] <- (ifelse(abs(tmpS[1,i]-qq[i])<1e-010,
                    mean(xx[tmpO[,i]]), xx[tmpO[1,i]]))
    }
    return(res)
}
```

Références

[1] Graf, M. (2002). Enquête suisse sur la structure des salaires 2000. Plan d'échantillonnage, pondération et méthode d'estimation pour le secteur privé. *Rapport de méthode 338-0010*, Office fédéral de la statistique, Neuchâtel.

[2] Särndal, C.-E.,Swensson, B. & Wretman, J. (1992). *Model Assisted Survey Sampling*. Springer Series in Statistics.

[3] R Development Core Team (2006). R : A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http ://www.R-project.org.

[4] Lumley, T. (2006). The survey Package. URL http://faculty.washington.edu/tlumley/survey/.

[5] Binder, D.A. (1991). Use of estimating functions for interval estimation from complex surveys. *Proceedings of the ASA Survey Research Methods Section* 1991 : 34-42.

Methodenberichte des Dienstes Statistische Methoden des BFS
Rapports de méthodes du Service de méthodes statistiques de l'OFS
Methodology reports published by the SFSO's Statistical Methods Unit

Ferrez, J., Graf, M. (2007). Enquête suisse sur la structure des salaires. Programmes R pour l'intervalle de confiance de la médiane. Numéro de commande : 338-0045

Renaud, A. (2007). Harmonisation de la scolarité obligatoire en Suisse (HarmoS). Design général de l'enquête et échantillon des écoles. Numéro de commande : 338-0044

Potterat, J. (2007). Betriebszählung 2005. Statistische Methoden zur Schätzung der provisorischen Ergebnisse. Bestellnummer : 338-0043

Hulliger, B. (2006). Umweltschutzausgaben der Unternehmen 2003, Stichprobenplan, Datenaufbereitung und Schätzverfahren. Bestellnummer : 338-0042

Renfer, J.-P. (2006). Enquête sur les chiffres d'affaires du commerce de détail. Plan d'échantillonnage et méthodes d'estimation. Numéro de commande : 338-0041

Salamin, P.-A. (2006). Statistique de l'aide sociale dans le domaine de l'asile. Plan de sondage et extrapolations pour l'enquête pilote 2005. Numéro de commande : 338-0040

Renaud, A. (2006). Statistique suisse des bénéficiaires de l'aide sociale. Pondération des communes 2004. Numéro de commande : 338-0039

Graf, M. (2006). Swiss Earnings Structure Survey 2002-2004. Compositional data in a stratified two-stage sample : Analysis and precision assessment of wage components. Order number : 338-0038

Potterat, J. (2006). Pensionskassenstatistik 2004. Statistische Methoden zur Schätzung der provisorischen Ergebnisse. Bestellnummer : 338-0037

Potterat, J. (2006). Kosten und Nutzen der Berufsbildung aus Sicht der Betriebe im Jahr 2004. Stichprobenplan, Gewichtung und Schätzverfahren. Bestellnummer : 338-0036

Kilchmann, D. (2006). Vierteljährliche Wohnbaustatistik. Stichprobenplan, statistische Datenaufarbeitung und Schätzverfahren 2005. Bestellnummer : 338-0035

Kilchmann, D. (2006). Erhebung über Forschung und Entwicklung in der schweizerischen Privatwirtschaft 2004. Bereinigung der Stichprobe, Ersatz fehlender Werte und Schätzverfahren. Bestellnummer : 338-0034

Kilchmann, D., Eichenberger, P., Potterat, J. (2005). Volkszählung 2000. Statistische Einsetzungsverfahren Band 2. Bestellnummer : 338-0033

Kilchmann, D., Eichenberger, P., Potterat, J. (2005). Volkszählung 2000. Statistische Einsetzungsverfahren Band 1. Bestellnummer : 338-0032

Graf, M., Matei, A. (2005). Enquête suisse sur la structure des salaires 2002. La précision du salaire brut standardisé médian. Numéro de commande : 338-0031

Graf, E., Renfer, J.-P. (2005). Enquête suisse sur la santé 2002. Plan d'échantillonnage, pondération et estimation de la précision. Numéro de commande : 338-0030

Potterat, J. (2005). Mietpreis-Strukturerhebung 2003. Gewichtung und Schätzverfahren. Bestellnummer : 338-0029

Potterat, J. (2005). Landwirtschaftliche Betriebszählung 2003. Schätzverfahren für die Zusatzerhebung. Bestellnummer : 338-0028

Renaud, A. (2004). Coverage estimation for the Swiss population census 2000. Estimation methodology and results. Order number : 338-0027

Kilchmann, D. (2004). Revision des Schweizerischen Lohnindex. Schätzmethoden der Lohnindices und deren Varianzschätzer. Bestellnummer : 338-0026

Graf, M. (2004). Enquête suisse sur la structure des salaires 2002. Plan d'échantillonnage et extrapolation pour le secteur privé. Numéro de commande : 338-0025

Renaud, A. (2004). Analyse de données d'enquêtes. Quelques méthodes et illustration avec des données de l'OFS. Numéro de commande 338-0024

Renaud, A., Potterat, J. (2004). Estimation de la couverture du recensement de la population de l'an 2000. Echantillon pour l'estimation de la sous-couverture (P-sample) et qualité du cadre de sondage des bâtiments. Numéro de commande : 338-0023

Graf, M. (2004). Fusion de données. Etude de faisabilité. Numéro de commande : 338-0022

Potterat, J. (2003). Mietpreis-Strukturerhebung 2003. Entwicklung des Stichprobenplans und Ziehung der Stichprobe. Bestellnummer : 338-0021

Potterat, J. (2003). Landwirtschaftliche Betriebszählung 2003. Stichprobenplan der Zusatzerhebung. Bestellnummer : 338-0020.

Renaud, A. (2003). Estimation de la couverture du recensement de la population de l'an 2000. Echantillon pour l'estimation de la sur-couverture (E-sample). Numéro de commande : 338-0019

Hulliger, B. (2003). Bereinigung der Stichprobe, Ersatz fehlender Werte und Schätzverfahren. Erhebung über F+E in der schweizerischen Privatwirtschaft 2000. Bestellnummer : 338-0018

Renfer, J.-P. (2003). Enquête 2000 sur la recherche et le développement dans l'économie privée en Suisse. Plan d'échantillonnage. Numéro de commande : 338-0017

Potterat, J. (2003). Kosten und Nutzen der Berufsbildung aus Sicht der Betriebe. Schätzverfahren. Bestellnummer : 338-0016

Graf, M., Matei, A. (2003). Stratégie de choix des modèles de désaisonnalisation. Application aux séries de l'emploi total. Numéro de commande : 338-0015

Potterat, J., Salamin, P.A. (2002). Betriebszählung 2001. Methoden für die Datenbereinigung. Bestellnummer : 338-0014

Renaud, A. (2002). Programme international pour le suivi des acquis des é lèves (PISA). Plans d'échantillonnage pour PISA 2000 en Suisse. Numéro de commande : 338-0013

Renfer, J.-P. (2002). Enquête 2001 sur les coûts et l'utilité de la formation des apprentis du point de vue des établissements. Plan d' échantillonnage. Numéro de commande : 338-0012

Potterat, J., Salamin, P.A. (2002). Betriebszählung 2001. Stichprobenplan und Schätzverfahren für die provisorischen Ergebnisse. Bestellnummer : 338-0011

Graf, M. (2002). Enquête suisse sur la structure des salaires 2000. Plan d'échantillonnage, pondération et méthode d'estimation pour le secteur privé. Numéro de commande : 338-0010

Renaud, A., Eichenberger P. (2002). Estimation de la couverture du recensement de la population de l'an 2000. Procédure d'enquête et plan d'échantillonnage de l'enquête de couverture. Numéro de commande : 338-0009

Kilchmann, D., Hulliger, B. (2002). Stichprobenplan für die Obstbaumzählung 2001. Bestellnummer : 338-0008

Graf, M. (2002). Passage du concept établissement au concept entreprise. Numéro de commande : 338-0007

Salamin, P.A. (2001). La technique de la double enquête pour la statistique du transport routier de marchandise. Numéro de commande : 338-0006

Peters, R., Renfer, J.-P. et Hulliger, B. (2001). Statistique de la valeur ajoutée 1997-1998. Procédure d'extrapolation des données. Numéro de commande : 338-0005

Potterat, J., Hulliger, B. (2001). Schätzung der Sägereiproduktion mit der Sägerei-Erhebung PAUL. Bestellnummer : 338-0004

Graf, M. (2001). Désaisonnalisation. Aspects méthodologiques et application à la statistique de l'emploi. Numéro de commande : 338-0003

Hüsler, J., Müller, S. (2001). Schlussbericht Betriebszählung 1995 (BZ 95), Mehrfach imputierte Umsatzzahlen. Bestellnummer : 338-0002

Renaud, A. (2001). Statistique suisse des bénéficiaires de l'aide sociale. Plan d'échantillonnage des communes. Numéro de commande : 338-0001

Hulliger, B., Eichenberger, P. (2000). Stichprobenregister für Haushalterhebungen : Umstellung auf Telefonnummern ohne Namen und Adressen, Abläufe für Erstellung und Stichprobenziehung. Bestellnummer : 338-0000

de Rossi, F.-X. (1998). Méthodes statistiques pour le compte routier suisse.

Hulliger, B., Kassab, M. (1998). Evaluation of Estimation Methods for the Survey on Environment Protection Expenditures of Swiss Communes.

Salamin, P.A. (1998). Etablissement d'une clef de passage pondérée entre l'ancienne (NGAE 85) et la nouvelle nomenclature (NOGA 95) générale des activités économiques.

Peters, R. (1998). Extrapolation des données de l'enquête de structure sur les loyers.

Bender, A., Hulliger, B. (1997). Enquête suisse sur la population active : rapport de pondération pour 1996.

Salamin, P.A. (1997). Evaluation de la Statistique de l'emploi.

Peters, R. (1997). Etablissement du plan d'échantillonnage pour l'enquête 1996 sur la recherche et le développement dans l'économie privée en Suisse.

Peters, R. (1997). Enquête 1996 sur la structure des salaires en Suisse : établissement du plan d'échantillonnage.

Peters, R. (1996). Pondération des données de l'enquête sur la famille en Suisse.

Comment, T., Hulliger, B., Ries, A. (1996). Gewichtungsverfahren für die Schweizerische Arbeitskräfteerhebung (1991-1995).

Hulliger, B. (1996). Haushalterhebung Familie 1994 : Stichprobenplan, Stichprobenziehung und Reservestichproben.

Peters, R., Hulliger, B. (1996). Schätzverfahren für die Lohnstruktur-Erhebung 1994 / Procédure d'estimation pour l'enquête de 1994 sur la structure des salaires.

Peters, R. (1996). Schéma de pondération des indices PAUL.

Hulliger, B., Peters, R. (1996). Enquête sur le comportement de la population suisse en matière de transport en 1994 : plan d'échantillonnage et pondération.

Hulliger, B. (1996). Gütertransportstatistik 1993 : Schätzverfahren mit Kompensation der Antwortausfälle.

Salamin, P.A. (1995). Estimation des flux pour le module II des comptes globaux du marché de travail.

Peters, R. (1995). Enquête de structure sur les loyers : établissement d'un plan d'échantillonnage stratifié.

Hulliger, B. (1995). Konjunkturelle Mietpreiserhebung : Stichprobenplan und Schätzverfahren.

Schwendener, P. (1995). Verbrauchserhebung 1990 - Vertrauensintervalle.

Peters, R., Hulliger, B. (1994). La technique de pondération des données : application à l'enquête suisse sur la santé.

Hulliger, B., Peters, R. (1994). Enquête sur la structure des salaires en Suisse : stratégie d'échantillonnage pour le secteur privé.

Publikationsprogramm BFS

Das Bundesamt für Statistik (BFS) hat – als zentrale Statistikstelle des Bundes – die Aufgabe, statistische Informationen breiten Benutzerkreisen zur Verfügung zu stellen.

Die Verbreitung der statistischen Information geschieht gegliedert nach Fachbereichen (vgl. Umschlagseite 2) und mit verschiedenen Mitteln

Programme des publications de l'OFS

En sa qualité de service central de statistique de la Confédération, l'Office fédéral de la statistique (OFS) a pour tâche de rendre les informations statistiques accessibles à un large public.

L'information statistique est diffusée par domaine (cf. verso de la première page de couverture); elle emprunte diverses voies:

Diffusionsmittel	*Kontakt* *N° à composer*	*Moyen de diffusion*
Individuelle Auskünfte	032 713 60 11 info@bfs.admin.ch	Service de renseignements individuels
Das BFS im Internet	www.statistik.admin.ch	L'OFS sur Internet
Medienmitteilungen zur raschen Information der Öffentlichkeit über die neusten Ergebnisse	www.news-stat.admin.ch	Communiqués de presse: information rapide concernant les résultats les plus récents
Publikationen zur vertieften Information (zum Teil auch als Diskette/CD-Rom)	032 713 60 60 order@bfs.admin.ch	Publications: information approfondie (certaines sont disponibles sur disquette/CD-Rom)
Online-Datenbank	032 713 60 86 www.statweb.admin.ch	Banque de données (accessible en ligne)

Nähere Angaben zu den verschiedenen Diffusionsmitteln liefert das laufend nachgeführte Publikationsverzeichnis im Internet unter der Adresse www.statistik.admin.ch→Aktuell→Publikationen.

La liste des publications, mise à jour régulièrement, donne davantage de détails sur les divers moyens de diffusion. Elle se trouve sur Internet à l'adresse www.statistique.admin.ch →Actualités→Publications.

Methodenberichte des Dienstes Statistische Methoden
Rapports de méthodes du Service de méthodes statistiques
Methodology Reports by the Statistical Methods Unit

Die Methodenberichte beschreiben die mathematischen und statistischen Methoden, die den Resultaten und Analysen der öffentlichen Statistik zu Grunde liegen. Sie enthalten ausserdem die Evaluation und Entwicklung von neuen Methoden im Hinblick auf eine zukünftige Anwendung. Diese Publikationen sollen einerseits die verwendeten Methoden dokumentieren, um Transparenz und Wissenschaftlichkeit sicher zu stellen, und sie sollen andererseits die Zusammenarbeit mit den Hochschulen und der Wissenschaft fördern.

Zur Illustration der beschriebenen mathematischen Konzepte, werden im Bericht numerische Resultate aufgeführt. Diese sind allerdings nicht als offizielle Resultate der betreffenden Erhebungen zu verstehen. Ebenfalls können die tatsächlich angewendeten Methoden leicht von den hier beschriebenen abweichen.

Die Methodenberichte sind auf der Internetseite des BFS in elektronischer Form verfügbar.

Les rapports de méthodes décrivent les méthodes mathématiques et statistiques à la base des résultats et des analyses de la statistique publique. Ils présentent également l'évaluation et le développement de nouvelles méthodes en vue d'une application future. Ces publications visent d'une part à documenter les méthodes utilisées ou envisagées dans un souci de transparence et de rigueur scientifique, et d'autre part à favoriser la collaboration avec le monde scientifique et universitaire.

Les résultats numériques présentés dans les rapports de méthodes illustrent les concepts mathématiques décrits, mais ne sont pas des résultats officiels des enquêtes concernées. De même, les méthodes réellement appliquées peuvent différer légèrement de celles décrites dans ces rapports.

Les rapports de méthodes sont disponibles sous forme électronique sur le site internet de l'OFS.

Ce rapport comporte deux parties. La première, plus mathématique, présente les calculs effectués dans le cadre de la LSE: la méthode suivie pour le calcul de la médiane, les différentes étapes nécessaires pour établir un intervalle de confiance à 95% et trois coefficients de variation ainsi que le traitement des domaines sont décrits. La deuxième partie décrit chaque élément du programme qui a été implémenté. Puis, les fonctions du package survey ayant un rapport avec les méthodes de la LSE sont analysées. Pour terminer, une comparaison des performances du programme et du package est présentée.

N° de commande: 338-0045

Commandes: 032 713 60 60

Fax: 032 713 60 61
E-Mail: order@bfs.admin.ch

Prix: gratuit

ISBN 978-3-303-00377-0